L Number	Hits	Search Text	DB	Time stamp
-	0	(client) and (server) and (request with access) and (session adj length) and (hash) and (authorization adj ticket) and ((shared adj secret) (secret adj shared)) and (request$3) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/17 08:35
-	0	(client) and (server) and (session adj length) and (hash) and (authorization adj ticket) and ((shared adj secret) (secret adj shared)) and (request$3) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 12:14
-	0	(client) and (server) and (session adj length) and (hash) and (ticket) and ((shared adj secret) (secret adj shared)) and (request$3) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 12:15
-	27262	(client) and (server) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 12:14
-	0	(client) and (server) and (session with length) and (hash) and (ticket) and ((shared adj secret) (secret adj shared)) and (request$3) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 12:15
-	14	(client) and (server) and (session with length) and (hash) and (ticket) and (request$3) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 12:15
-	14	(client) and (server) and ((session connection) with length) and (hash) and (ticket) and (request$3) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 12:18
-	14	(client) and (server) and ((session connection) with length) and (hash) and (ticket) and (request$3 with authorization) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 12:18
-	0	(client) and (server) and ((session connection) with length) and (hash) and (authorization with ticket) and (request$3 with authorization) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 12:18
-	0	(client) and (server) and ((session connection) with length) and (authorization with ticket) and (request$3 with authorization) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 12:19
-	0	(client) and (server) and (authorization with ticket) and (request$3 with authorization) and ((session connection) with lenght (length with time)) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 12:20
-	30	(client) and (server) and (authorization with ticket) and (request$3 with authorization) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 12:46
-	0	(hash$3 encrypt$3) same (ticket cookie certificate token) same ((session connection) adj length) same ((secret adj shared) (shared adj secret)) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 12:49
-	1171	((session connection) adj length) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 12:49
-	4	(hash$3 encrypt$3) same ((session connection) adj length) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 17:07
-	2	(authorization) same ((session connection) adj length) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 12:55

-	8	(authorization) same (((session connection time) adj length)) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 13:08
-	0	(hash$3) same (ticket token cookie certificate) same (((session connection time) adj length)) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 13:09
-	1	(hash$3) and (ticket token cookie certificate) same (((session connection time) adj length)) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 13:09
-	30	(ticket token cookie certificate) same (((session connection time) adj length)) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 13:21
-	4	(ticket token cookie certificate) same (((session) adj length)) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 13:23
-	21	(ticket token cookie certificate) and (((session) adj length)) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 13:43
-	276	(ticket token cookie certificate) and (timestamp "time stamp") and (elapsed) and ("time counter" clock) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 13:44
-	30	(ticket token cookie certificate) same (timestamp "time stamp") and (elapsed) and ("time counter" clock) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 13:47
-	2	(ticket token cookie certificate) and (timestamp "time stamp") and (elapsed) and ("time counter" clock) and (session with length) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 13:49
-	530	(md5) and (hash) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 13:50
-	52	(md5) same (hash) and ticket and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 13:50
-	2	(md5) same (hash) same ticket and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 13:55
-	10	(persistence adj timer) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 14:02
-	75	(web adj server) and (instant with (messenger messag$3)) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 14:04
-	22	(web adj server) and (instant with (messenger messag$3)) and (ticket token cookie certificate) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 14:13
-	4	(web adj server) and (instant with (messenger messag$3) with server) and (ticket token cookie certificate) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 14:19
-	50	(timestamp "time stamp") same (elapsed "session length") same (current adj time) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 14:22
-	0	(timestamp "time stamp") same (elapsed "session length") same (current adj time) same (threshold) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 14:23

-	15	(timestamp "time stamp") same (elapsed "session length") same (current adj time) and (threshold) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 14:27
-	49	(timestamp "time stamp") same (elapsed "session length") same (current adj time) and (add$3 sum$4) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 14:28
-	3	(timestamp "time stamp") same (elapsed "session length") same (current adj time) same (add$3 sum$4) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 14:32
-	10	(timestamp "time stamp") same (elapsed (session connection login logon) with (length time)) same (current adj time) same (add$3 sum$4) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 14:49
-	21	(timestamp "time stamp") same (current adj time) same (add$3 sum$4) same (compar$3) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 14:52
-	9	(timestamp "time stamp") with (add$3 sum$4) same (current adj time) with (compar$3) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 14:52
-	805	(ticket) with (secret key) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 17:07
-	8	(ticket) with (shared adj (secret key)) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/16 17:07
-	724	(session connection) same (timer) same (length) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/17 08:35
-	105	(session connection) with (timer) with (length) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/17 08:35
-	17	(session) with (timer) with (length) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/17 09:19
-	1682	(hash$3) and (ticket token cookie) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/17 09:20
-	293	(hash$3) same (ticket token cookie) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/17 09:20
-	168	(hash$3) same (ticket token cookie) and ((ticket token cookie) same (time)) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/17 09:21
-	0	(hash$3) same (ticket token cookie) and ((ticket token cookie) same (sesion with time)) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/17 09:21
-	18	(hash$3) same (ticket token cookie) and ((ticket token cookie) same (session with time)) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/17 09:31
-	578	((ticket token cookie) same ((session connection login logon) with (elapsed time period length))) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/17 09:33
-	316	((ticket token cookie) with ((session connection login logon) with (elapsed time period length))) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/17 09:46

-	180	((ticket token cookie) with ((active) with (elapsed time period length))) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/17 09:46
-	163	((ticket token cookie) with ((active) with (time period))) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/17 15:41
-	34	(receiv$3 with (acknowledg$3)) same (timer) same (compar$3) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/17 16:25
-	1	(receiv$3 with (acknowledg$3)) same (timer) same (compar$3) same (delet$3) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/17 16:16
-	8	(delet$3) same (ticket) same (invalid$5) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/17 16:17
-	30	receiv$3 same (timer) same (compar$3) same (delet$3) and @ad<20000829	USPAT; US-PGPUB; EPO; JPO; IBM_TDB	2004/03/17 16:26

US005999711A

United States Patent [19]

Misra et al.

[11] Patent Number: 5,999,711

[45] Date of Patent: Dec. 7, 1999

[54] **METHOD AND SYSTEM FOR PROVIDING CERTIFICATES HOLDING AUTHENTICATION AND AUTHORIZATION INFORMATION FOR USERS/MACHINES**

[75] Inventors: **Pradyumna K. Misra**, Issaquah; **Arnold S. Miller**, Bellevue; **Richard B. Ward**, Seattle, all of Wash.

[73] Assignee: **Microsoft Corporation**, Redmond, Wash.

[21] Appl. No.: **08/277,144**

[22] Filed: **Jul. 18, 1994**

[51] **Int. Cl.**[6] **G06F 11/00**; G06F 13/14

[52] **U.S. Cl.** **395/187.01**; 395/186; 395/680; 395/200.09; 308/25; 308/4

[58] **Field of Search** 395/187.01, 186, 395/680, 200.09; 380/4, 23, 25

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,896,319	1/1990	Hidinsky et al.	370/60
4,993,068	2/1991	Piosenka et al.	380/23
5,224,163	6/1993	Gasser et al.	380/30
5,235,642	8/1993	Wobber et al.	380/25
5,335,346	8/1994	Fabbio	395/600
5,560,008	9/1996	Johnson et al.	395/650

FOREIGN PATENT DOCUMENTS

421409A2	4/1991	European Pat. Off.	G07F 7/10
2238636A	6/1991	United Kingdom	G06F 1/00

OTHER PUBLICATIONS

Steiner, Jennifer G. et al., "Kerberos: An Authentication Service for Open Network Systems," in USENIX Winter Conference Proceedings, Feb. 9–12, 1988, Dallas, Texas, pp. 191–202.

Neuman, Clifford B., "Proxy-Based Authorization and Accounting for Distributed Systems," Department of Computer Science and Engineering, University of Washington, Technical Report 91–02–01, Mar., 1991, pp. 1–14.

Karger, Paul A., and Andrew J. Herbert, "An Augmented Capability Architecture to Support Lattice Security and Traceability of Access," in Proceeding of the 1984 Symposium on Security and Privacy, Apr. 29–May 2, 1984, Sponsored by the Technical Committee on Security and Privacy IEEE Computer Society, pp. 2–12.

Saltzer, Jerome H., and Michael D. Schroeder, "The Protection of Information in Computer Systems," in Proceedings of the IEEE 63(9), Sep., 1975, pp. 1278–1308.

Israel, Jay E., and Theodore A. Linden, "Authentication in Office System Internetworks," *ACM Transactions on Office Information Systems 1*(3), Jul.,1983, pp. 193–210.

Ciminiera, L., and A. Valenzano, "Efficient Authentication Mechanisms Using the iAPX–432," *Interfaces in Computer 3*, 1985, pp. 111–124.

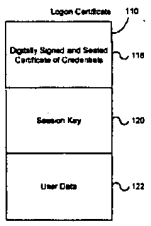
(List continued on next page.)

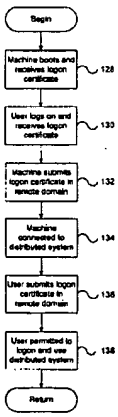
Primary Examiner—Hoa T. Nguyen
Attorney, Agent, or Firm—Seed and Berry LLP

[57] **ABSTRACT**

Logon certificates are provided to support disconnected operation within the distributed system. Each logon certificate is a secure package holding credentials information sufficient to establish the identity and rights and privileges for a user/machine in a domain that is not their home domain. When a user/machine attempts to connect to the system at a domain other than the home domain of the user/machine, the user/machine presents a logon certificate that evidences his credentials. The domain where the user/machine attempts to connect to the system, decrypts and unseals the secure package as required to obtain the credentials information contained therein. If the user/machine has sufficient credentials, the user/machine is permitted to connect to the system. If the user/machine lacks sufficient credentials, the user/machine is not permitted to connect to the system.

16 Claims, 6 Drawing Sheets





OTHER PUBLICATIONS

Pinkas, Denis, "An Access Control Model for Distributed Systems Based on the Use of Trusted Authorities," in SECURICOM. 7th Worldwide Congress on Computer and Communications Security and Protection, 1989, pp. 257–270.

Kohl, John, and Clifford Neuman, Kerberos Version 5 RFC, Revision #5, Memorandum of Apr. 9, 1992, pp. 1–68.

Muftic, Sead, and Morris Sloman, "Security architecture for distributed systems," *Computer Communications*, 17(7): 492–500, Jul., 1994.

Bacon et al., "Extensible Access Control for a Hierarchy of Servers," *Operating Systems Review* 28(3):4–15, 1994.

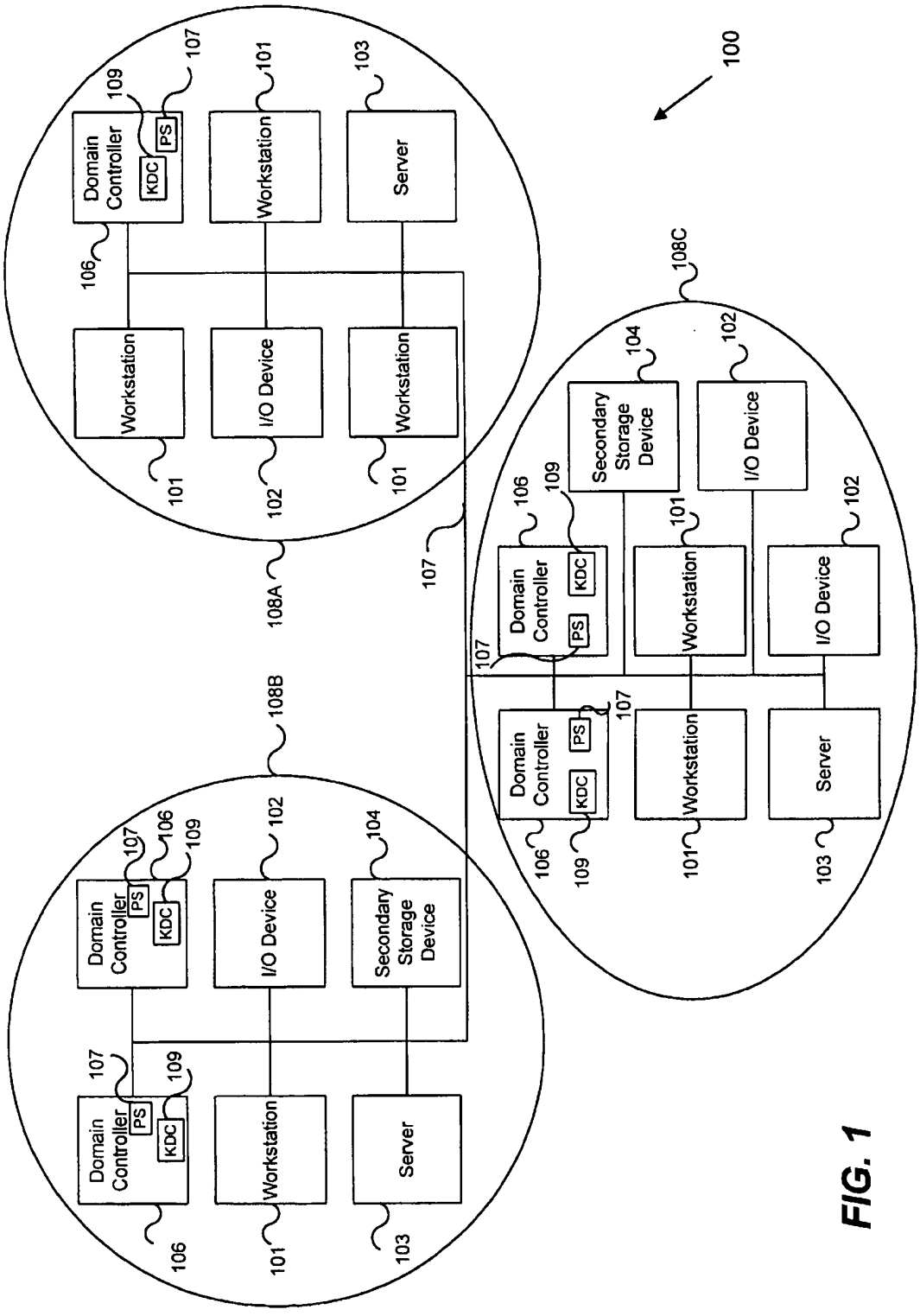

100
108A
108B
108C
Domain Controller
KDC
PS
Workstation
Server
I/O Device
Secondary Storage Device
101
102
103
104
106
107
109

FIG. 1

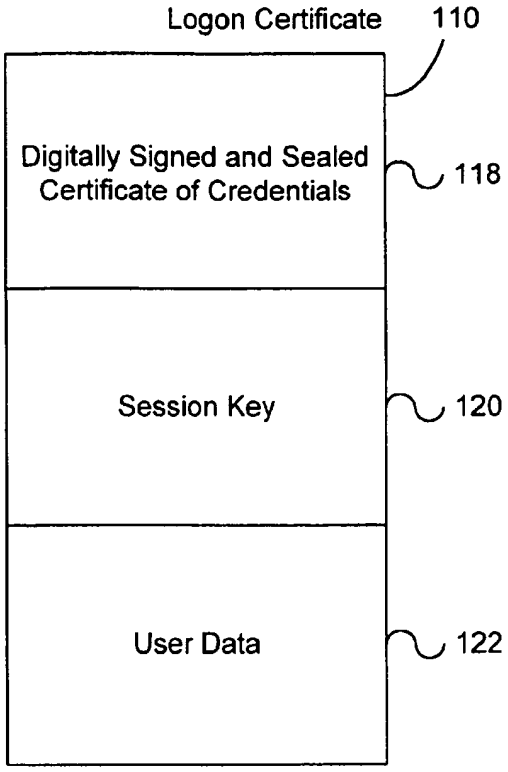

Logon Certificate
110
Digitally Signed and Sealed Certificate of Credentials
118
Session Key
120
User Data
122

FIG. 2A

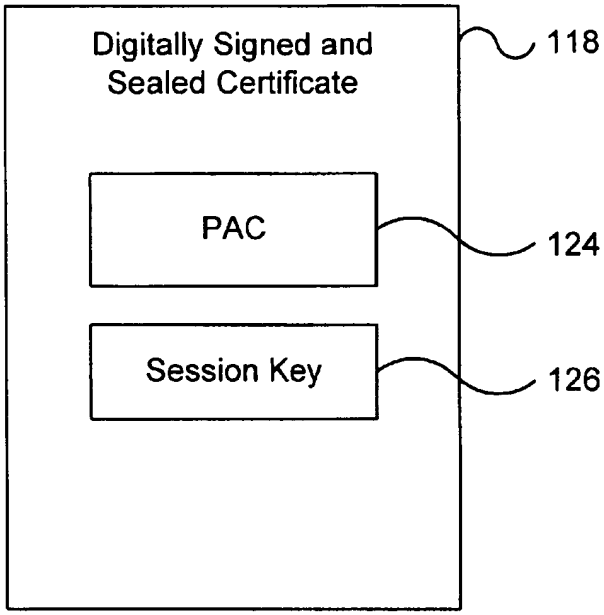

Digitally Signed and Sealed Certificate
118
PAC
124
Session Key
126

FIG. 2B

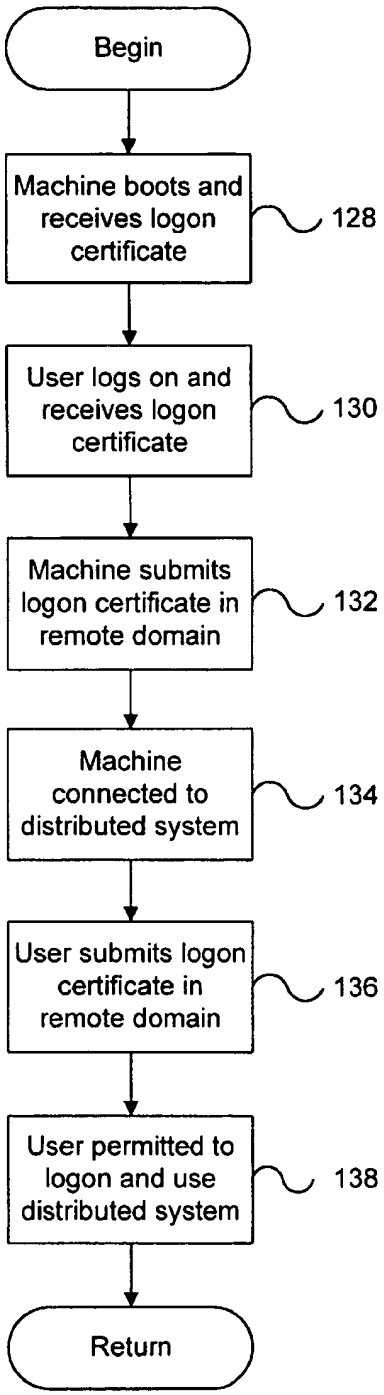

Begin
Machine boots and receives logon certificate
128
User logs on and receives logon certificate
130
Machine submits logon certificate in remote domain
132
Machine connected to distributed system
134
User submits logon certificate in remote domain
136
User permitted to logon and use distributed system
138
Return

FIG. 3

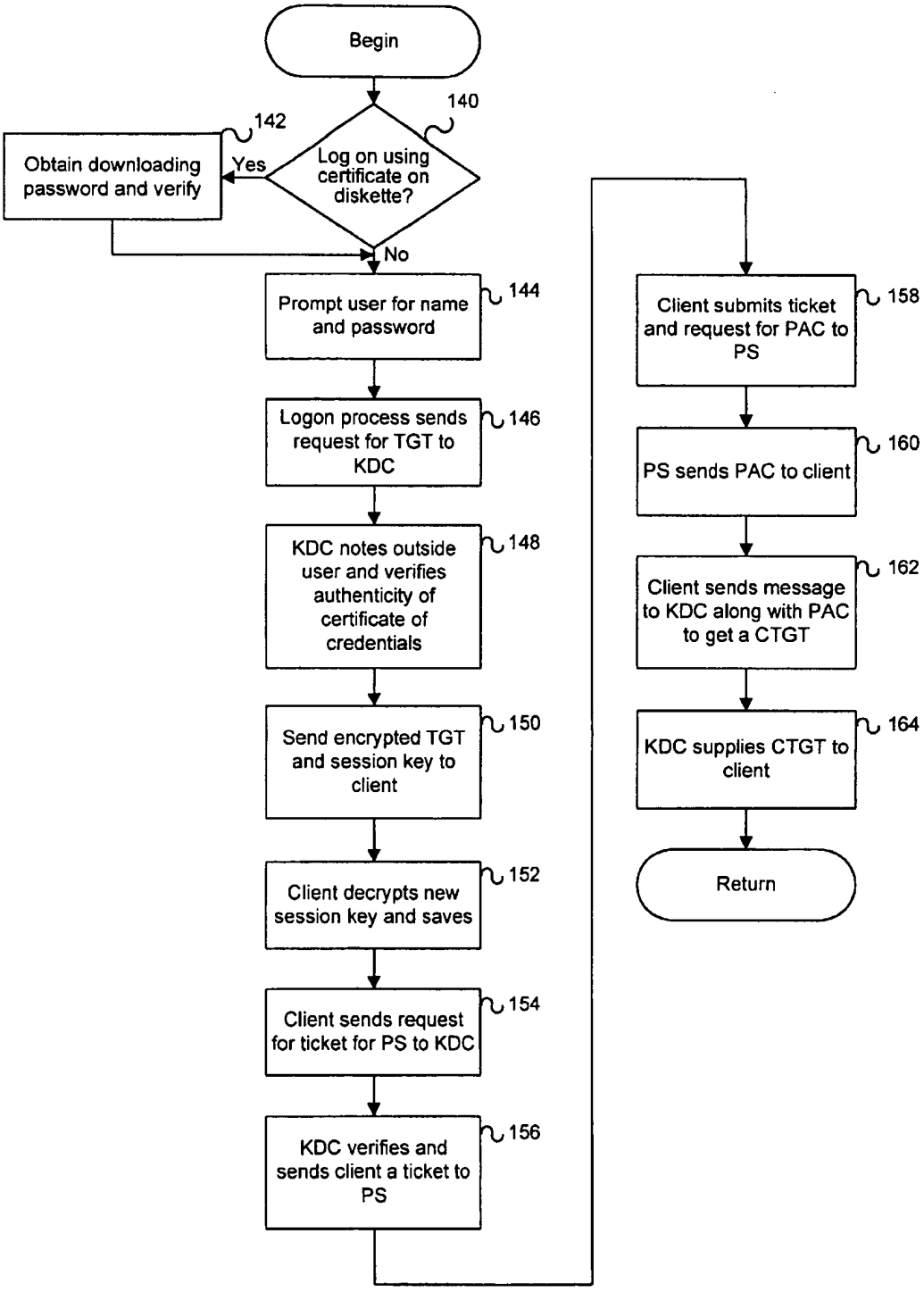

Begin
140
Log on using certificate on diskette?
142
Yes
Obtain downloading password and verify
No
144
Prompt user for name and password
146
Logon process sends request for TGT to KDC
148
KDC notes outside user and verifies authenticity of certificate of credentials
150
Send encrypted TGT and session key to client
152
Client decrypts new session key and saves
154
Client sends request for ticket for PS to KDC
156
KDC verifies and sends client a ticket to PS
158
Client submits ticket and request for PAC to PS
160
PS sends PAC to client
162
Client sends message to KDC along with PAC to get a CTGT
164
KDC supplies CTGT to client
Return

FIG. 4A

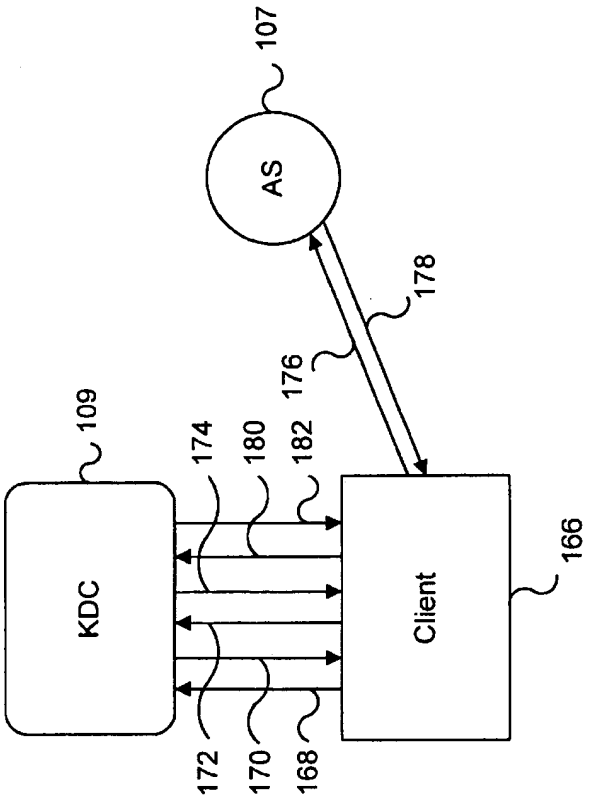


FIG. 4B

METHOD AND SYSTEM FOR PROVIDING CERTIFICATES HOLDING AUTHENTICATION AND AUTHORIZATION INFORMATION FOR USERS/MACHINES

TECHNICAL FIELD

The present invention relates generally to data processing systems and, more particularly, to the use of logon certificates in a distributed system.

BACKGROUND OF THE INVENTION

Many conventional distributed systems do not support roaming users or roaming machines. A roaming user may wish to logon to the distributed system at domains other than his home domain. Similarly, roaming machines may wish to connect to the distributed system at sites outside of their home domain. The roaming user may use a roaming machine (e.g., a portable computer) to logon or may instead use a connected computer that is available at the logon site. The conventional systems that have supported such roaming users and machines have provided the support at the expense of efficiency and increased vulnerability. For example, certain conventional distributed systems store credentials information at a home domain of the user/machine. The credentials information stored at the home domain is examined when the user/machine tries to connect to the system at a different domain. The credentials information is examined to determine whether the user/machine is permitted to connect to the distributed system. In order to facilitate roaming users and machines, these conventional distributed systems replicate the credentials information to each potential connection domain in the distributed system (i.e., to each domain).

This approach of replicating credentials across the system suffers from several drawbacks. First, the replication of the credentials information is costly and time-consuming. Second, the credentials information must be replicated frequently because credentials must be updated each time that the credentials information of any user or machine changes. Third, the replication of credentials may not be successful due to intermittent failure, and, thus, the proper credentials information may not reach all the targeted destinations in the distributed system. Fourth, this approach poses a security threat because it provides more locations within the distributed system that are susceptible to attack.

SUMMARY OF THE INVENTION

The drawbacks of the conventional systems are overcome by the present invention. In accordance with a first aspect of the present invention, a method is practiced in the distributed system that has a facility for checking authorization and authentication information, typically referred to as credentials. In this method, a principal, such as a user or portable computer, is provided with a secure package that holds certified credential information for the principal. The secure package may be encrypted and/or may include a digital signature. The secure package may be provided to the principal by storing the secure package on a portable storage medium such as a floppy disk. Alternatively, the secure package may be provided to the principal by storing the secure package in the memory of a portable computer of the user.

Once the principal has been provided with a secure package, the principal may send a request to logon to the distributed system along with the certificate of credentials that is received by the distributed system. The secure package is accessed to enable the facility for checking credentials to determine whether the principal is authorized to connect to the distributed system. Where the principal is not authorized to connect to the distributed system, the principal's request to logon is denied. In contrast, where the principal is authorized to connect to the distributed system, the principal's request to connect is granted.

In accordance with another aspect of the present invention, a distributed system is logically partitioned into domains. Each user has an associated home domain. Authorization and authentication information about a user is encrypted to produce a block of encrypted credentials information. A digital signature is attached to the encrypted credentials information at the home domain for the user. The digital signature is created using a private key for the home domain. A session key is received from the user and is used to encrypt the digital signature and the block of encrypted credentials information to produce a secure package. The secure package is provided to the user to enable the user to logon to the distributed system in a domain other than the home domain.

The digital signature may be created by using a hash function to generate a hash value of the credentials information. An encryption key is selected to bulk encrypt the credentials information and then the hash value and the selected encryption key together are encrypted using the private key of the home domain. The resulting product is the digital signature.

In accordance with an additional aspect of the present invention, the method is practiced at a distributed system that has a facility for verifying and validating the credentials information. A portable computer is provided with a secure package that holds the credentials information for the computer. The portable computer is required to present the secure package when it wishes to connect to the distributed system in a domain other than its home domain. A facility for verifying and validating the credentials information examines the credentials contained within the secure package to determine whether the computer is authorized to connect to the distributed system. If the portable computer is authenticated, it is permitted to connect to the distributed system. On the other hand, where the portable computer is not authenticated, the portable computer is not allowed to connect to the distributed system.

In accordance with yet another aspect of the present invention, a method is practiced in the distributed system that includes a plurality of computers and is logically partitioned into domains. Each computer in the distributed system has an associated home domain. A secure package is provided at the home domain of the computers. A secure package holds credentials information for the selected computer. A request is received from the selected computer to connect to the distributed system at a target domain that lies outside the home domain of the selected computer. The secure package is received from the selected computer and the credentials information contained within the secure package is examined to determine whether the selected computer is authorized to be connected to the distributed system at the target domain.

In accordance with a further aspect of the present invention, a user is provided first with an option of logging on to the distributed system interactively. In this first option, a certificate of credentials is not required. The user is also provided with a second option of logging on to the distributed system wherein a certificate of credentials is required. When a request to logon using the second option is received

from a user, it must be accompanied by a certificate of credentials. The certificate of credentials for the user is examined to determine whether the user has sufficient credentials to be permitted to logon. Where it is determined that the user has sufficient credentials, the user is permitted to logon. On the other hand, where it is determined that the user lacks sufficient credentials, the user is prohibited from logging on.

In accordance with a still further aspect of the present invention, a secure certificate of credentials is provided to a user to allow the user to logon to the distributed system outside of the associated home domain for the user. A start time is established for the certificate of credentials that determines when this certificate becomes valid. A time of expiration is established for the certificate of credentials. Once the time of expiration has passed, the user cannot logon using the expired certificate of credentials and any attempt by user to logon using the certificate of credentials to the distributed system is rejected by the distributed system.

In accordance with an additional aspect of the present invention, when a request is received from a user to receive a secure certificate of credentials that will enable the user to logon to the distributed system outside its associated home domain, the user is prompted for a password. The information indicative of the password is encoded into the certificate of credentials before the certificate of credentials is issued to the user. The user is required to show knowledge of the password when attempting to logon using the certificate of credentials.

In accordance with yet another aspect of the present invention, a certificate of credentials is issued to a user to enable the user to logon to the distributed system using the certificate of credentials to show that the user has proper and sufficient credentials. It is also possible to revoke the certificate of credentials so that the certificate of credentials may no longer be used to logon to the distributed system.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a distributed system that is suitable for practicing a preferred embodiment of the present invention.

FIG. **2A** is a diagram illustrating the contents of a logon certificate in accordance with the preferred embodiment of the present invention.

FIG. **2B** is a diagram illustrating in more detail the contents of the digitally signed and sealed certificate of FIG. **2A**.

FIG. **3** is a flow chart illustrating the steps performed to obtain and use logon certificates in the preferred embodiment of the present invention.

FIG. **4A** is a flow chart illustrating the steps that are performed for a user to logon to the distributed system using a logon certificate in accordance with the preferred embodiment of the present invention.

FIG. **4B** is a block diagram illustrating the interaction between major components of the distributed system during a logon in accordance with the preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The preferred embodiment of the present invention provides a secure and efficient approach for supporting roaming users and roaming machines in a distributed system environment. In particular, the preferred embodiment of the present invention issues secure logon certificates to users/machines which may be later presented to the distributed system at locations other than the home domains of the users/machines to enable connection to the distributed system. The logon certificates encapsulate credentials information for the users/machines. Since each user/machine carries the credentials directly to the desired connection point i.e. the domain, the credentials information held in the logon certificate need not be replicated throughout the distributed system and there is no need to establish a direct connection back to the domain controller of the home domain. Moreover, the logon certificates provide a convenient vehicle for controlling access to the system. Logon certificates are revocable and expire after a predetermined period of time. Further, encryption techniques are applied to contents of the logon certificate to ensure that the contents are secure.

FIG. **1** is a block diagram of a distributed system **100** that is suitable for practicing the preferred embodiment of the present invention. Those skilled in the art will appreciate that the configuration shown in FIG. **1** is merely illustrative and that the present invention may be practiced in other distributed system configurations. The distributed system **100** of FIG. **1** includes workstations **101**, input/output (I/O) devices **102**, network servers **103** and secondary storage devices **104**. In addition, the distributed system **100** includes domain controllers **106** (which will be described in more detail below).

The distributed system **100** is logically partitioned into domains **108A**, **108B** and **108C**. Each domain **108A**, **108B** and **108C** is a self-sufficient collection of resources that is viewed as a single entity for purposes of administration, naming and security. Each domain implements its own administrative and security policies. Domains are provided to facilitate scaling and encapsulation of resources into logical units within the distributed system.

Although the preferred embodiment of the present invention utilizes domains, those skilled in the art will, nevertheless, appreciate that domains are not a necessary component for practicing the present invention. The present invention may be practiced in environments that do not utilize domains.

Each domain **108A**, **108B** and **108C** includes at least one domain controller **106**. More than one domain controller **106** may be provided within a domain so as to enhance the availability of domain controller resources. Each domain controller **106** serves as a centralized location for storing knowledge about the namespace of the distributed system **100**. Among the components included within each domain controller **106** are an authorization service (AS) **107** and an authentication service, known as the key distribution center (KDC) **109**. The AS **107** is a service that controls authorization rights that are provided to clients and validates requests to gain access to servers. A client, in this context, is a process that makes use of a network service on behalf of a user. The KDC **109** acts as an authentication service in that it authenticates the identities of principals. A principal is a uniquely named client or server instance. A server is a principal that provides a resource to clients. The AS **107** and KDC **109** work in conjunction to perform authorization and authentication respectively when a user wishes to logon to the system **100** and use its resources.

It is helpful to first look at the authentication performed during a logon. For each user/machine, the user/machine engages in an authentication protocol, such as the Kerberos,

version 5, release 5 protocol developed by the Massachusetts Institute of Technology, which will be described in more detail below. For the present invention, we are concerned solely with the authentication that is performed when a user/machine seeks to connect to the system outside its home domain. Through this protocol, users and machines exhibit knowledge of shared secrets that serve as credentials that verify the identity of the user/machine.

Each domain controller **106** holds information about users and machines for which the domain is the home domain. The domain controller **106** of a domain holds both the authentication credentials and authorization information for each of its users and machines. Logon certificates provide a vehicle to demonstrate that the user/machine has sufficient credentials to connect to the non-home domain without contacting the home domain. Thus, a user/machine may freely roam the distributed system and connect to all domains where such connection is authorized. This approach, however, does not require replication of the credentials throughout the distributed system, as required by conventional systems.

The logon certificates may be created in part through the use of encryption mechanisms. In particular, the logon certificates may be made secure by using an asymmetric encryption strategy wherein a public key and a private key pair is utilized to encrypt portions of the logon certificates. A noteworthy characteristic of an asymmetric encryption scheme is that it allows different keys for encryption and decryption unlike symmetric encryption systems that use the same key for both encryption and encryption. These keys are referred to as public and private key pair. Each domain has an associated key pair that includes a public key and a private key that are used in encryption as will be described in more detail below. The public key is published exclusively to domain controllers **106** in other domains through a location and distribution protocol provided by the KDC **109** of the domain controller **106**. The publication occurs during domain installation without user involvement. Thus, for instance, a domain controller **106** in domain **108**A publishes its public key to domain controllers **106** in domains **108**B and **108**C. So as to enhance the integrity of the system, the public keys are only published to the domain controllers **106** in other domains and not to other entities. Hence, the public keys are not made entirely public knowledge. The private key, in contrast, is kept secret within the domain controller **106** of the domain.

FIG. **2**A is a block diagram showing the components of a logon certificate **110**. A logon certificate **110** is a sealed packet that includes a digitally signed and sealed certificate of credentials **118**, a session key **120** and, optionally, user or machine specific data **122**. The logon certificate **110** also contains information such as the time the logon certificate was issued, the time of expiration of the logon certificate and the time at which the logon certificate becomes valid (i.e., start time).

The digitally signed and sealed certificate **118** includes at least (as shown in FIG. **2**B) a privileged attribute certificate (PAC) **124** and another copy of the session key **126**. The PAC **124** encapsulates authorization information for a user or machine. For instance, a user's security ID and the security ID of all the groups of which the user is a member are included in the PAC for a user, along with other authorization information, such as the user's privileges and the like. Only authentication information for a machine is encapsulated in a logon certificate for the machine. In the preferred embodiment of the present invention, the digitally signed and sealed certificate **118** is created by initially generating a hash of the contents to be included within the digitally signed and sealed certificate. A one-way hash function, such as the MDS hash function proposed by Ron Rivest, is used to generate the hash of the contents of the digitally signed and sealed certificate **118**. A random encryption key is then selected and the data to be included in the certificate **118** is encrypted using the randomly selected key using a bulk encryption algorithm like the RC4 encryption algorithm proposed by Ron Rivest. Lastly, a digital signature is created by encrypting the pair of the hash and the encryption key together by the domain's private key for the domain issuing the certificate using an asymmetric encryption algorithm like the RSA encryption algorithm. Note that use of MD5, RC4, and RSA is not a requisite to practicing this invention and comparable algorithms may be substituted for these algorithms.

The session key **126** is a dynamically generated key that is particular to a given communications session. The session key may be used to encrypt communications during a session between a client and a server, such as between a user's machine and the KDC 109.

The logon certificate **110** that includes a session key, a digital signature, a sealed certificate of credentials, and other information, like issuing domain, etc., is sealed by further encrypting it with a user-supplied password. The domain where the logon certificate **110** is later used will have been a recipient of the public decryption key that was distributed. If this domain has been configured to accept certificates from the issuing domain, it will decrypt the logon certificate in order to recover its contents.

Those skilled in the art will appreciate that the contents of the logon certificate **110** may include additional information or different information from that shown in FIGS. **2**A and **2**B. The information depicted in FIGS. **2**A and **2**B constitutes what is included in the preferred embodiment of the present invention and is not intended as a limitation of the scope of the present invention as defined in the appended claims.

FIG. **3** is a flow chart illustrating the basic steps involved in using the logon certificates in the preferred embodiment of the present invention. The sequence of steps is intended to merely be illustrative, and the steps may be performed in a different order. Moreover, all the steps need not be performed to practice the present invention.

Every time a machine is booted in its home domain it obtains a logon certificate **110** from the domain controller of its home domain that certifies the identity of the machine (step **128**). The certificate is obtained and issued if and only if machine was able to authenticate itself to the distributed system. The machine can later submit the logon certificate **110** to the domain controller **106** of another domain to which it is being connected when the machine boots. Similarly, every time a user logs on in his home domain, he obtains a logon certificate **110** from the domain controller **106** (step **130** in FIG. **3**). Again, the certificate is issued if and only if the user was able to authenticate himself to the distributed system. The logon certificate **110** may then later be used to logon at a site in a different domain. The non-home domain to which the user is permitted to logon using the logon certificate **110** is known as the "connection domain" and provides connectivity services to the user.

A user may request to download the logon certificate **110** onto a removal storage media, such as floppy diskette. When the user requests to download a logon certificate **110** onto such a removable storage media, he is prompted to supply a password. A one-way hash function is used to hash this password, which is then used to generate an encryption key,

which in turn is used to further encrypt the logon certificate. This password is required to prevent any third party that is in possession of a logon certificate on a removable storage media from fraudulently logging on as the user to whom the logon certificate was issued.

Once the machine and user have received logon certificates **110**, typically, the user tries to boot the machine at a remote domain within the distributed system **100**. During the boot of the machine at the remote domain, the machine submits the logon certificate **110** to the distributed system **100** to identify itself and to verify that the machine is authorized to be connected to the distributed system (step **132** in FIG. **3**). If the machine submits the proper credentials, the machine is then connected to the distributed system **110** (step **134**). The user may then logon to the distributed system. As part of the logon process (as will be described in more detail below), the user submits the logon certificate **110** (step **136**). If the logon certificate **110** indicates that the user is an authorized user, the user is permitted to logon and use the distributed system (step **138**).

It should be appreciated that the steps shown in the flow chart of FIG. **3** are for an instance wherein the user is carrying a portable machine for logging on at a remote domain. There may be instances wherein the user, instead, utilizes a machine that is already connected to the distributed system **100** at a remote domain. In such an instance, the user may be required to present a logon certificate **110** to logon at the remote domain, but there may be no logon certificate on the machine the user is using. Alternatively, there may be instances wherein a portable computer is used by a user at his home domain. In such an instance, the portable computer may present a logon certificate **110** to get connected to the distributed system **100** but the user need not present such a logon certificate.

In order to understand how the logon certificates **110** are used in logging on to the distributed system **100**, it is helpful to review some of the fundamentals of the Kerberos protocol. Kerberos uses "tickets" to regulate access by clients to servers. A ticket is a data structure, such as a record, that holds data like the target server name, client name and authorization data, that helps a client to authenticate itself to a server. The ticket allows a client to receive service from a server.

An authenticator is a data structure that includes the client's name, time stamp, and may also include an optional session key. The authenticator includes data that is encrypted in the session key. This data evidences that the sender knows the session key. Further, the time stamp helps to minimize the time period in which an eavesdropper may use a copied ticket and authenticator pair.

In Kerberos, session keys are used to verify the credentials of clients and may also be used to encrypt messages between two parties (e.g., a client and a server) during a given communication session. When the communication session ends, the session key is destroyed. The session key is shared only between the two parties that utilize it.

Kerberos maintains an authentication database that correlates clients, such as users, with their associated secret keys. As was mentioned above, a secret key is typically an encrypted password or is derived from one using a prespecified algorithm. This authentication database is utilized to retrieve the secret keys of the clients as needed during execution of the protocol.

As mentioned above, logon certificates **110** are used when a user attempts to logon to the distributed system **100**. FIG. **4A** shows a flow chart of the steps that are performed in such an instance. The steps of the flow chart of FIG. **4A** will be described below in conjunction with the block diagram of the system components shown in FIG. **4B** (which depicts the major components that play a role in authorizing and authenticating the user). When a roaming user attempts to logon, he is presented with a logon menu that includes an option to "logon via certificate." The logon menu is provided as part of a local logon process that serves as a client for the user. If the user selects this option, he may wish to logon utilizing a logon certificate **110** carried on a removable storage media, such as a floppy diskette or use the logon certificate stored on the machine he is attempting to logon from. Thus, in step **140**, it is determined whether the user is logging on using a certificate on a removable storage media or a certificate stored in the user's machine for the user. If the user decides to logon using a logon certificate **110** that is contained on a removable storage media, the user must provide the downloading password that he was required to enter when downloading the certificate onto the removable storage media and the system verifies that this is the correct downloading password (step **142**). Failure to supply the correct downloading password will prevent the user from using the logon certificate **110** stored on the removable storage media. If the user wishes to use the logon certificate stored on the machine (most likely the portable computer the user uses often), then he must supply his normal logon password.

In either case the logon process uses this password to generate an encryption key using a pre-specified and fixed algorithm. It then uses this key to decrypt the logon certificate retrieved from either the removable storage media or the machine itself. The client thus obtains an encrypted session key that was stored in the encrypted logon certificate and is also stored in the encrypted block of credentials. The client will use this encrypted session key exactly in the way it would use the key derived from the user-supplied password in case of a normal logon sequence without using logon certificate.

The system initiates an authentication exchange by prompting the user for the user's name and password (step **144**). The logon process then sends a request for a ticket granting ticket (TGT) to KDC **109**, which runs on a domain controller **106** in the local domain (step **146**). The logon certificate **110** is sent along with the request for TGT. The request for the TGT is represented by arrow **168** in FIG. **4B**. The KDC **109** determines that the logon request is by a user in some other domain and then looks up the public key associated with the domain specified in the logon certificate. It uses this public key to unseal the certificate of credentials and verify that the certificate was indeed issued by the domain named in the un-encrypted part of the logon certificate **110** and that the logon certificate is valid (i.e. it has not expired) and that the current time is past the start time held in the certificate. The KDC **109** also obtains the encrypted session key from the sealed certificate of credentials **118** and uses it exactly in the same way as it would have used the encryption key derived from the one way hash of user's password stored in its database for the users it has entries for in its authentication database. Once the KDC **109** has verified the authenticity of the certificate **118** by verifying the digital signature and concluded that the certificate is valid (step **148**), the KDC **109** then sends a TGT and a new session key (to be used in further communications) that has been encrypted by the session key obtained from the sealed certificate, to the client **166** (step **150**) as represented by arrow **170**.

The client **166** decrypts the new session key sent by the KDC(since the client is in possession of the session key from

decrypting the logon certificate as described above) and saves the new session key for future use (step **152**). The client **166** then initiates a request for a credentials ticket granting ticket (CTGT) by asking the KDC **109** for a service ticket to the AS **107** (step **154**). The CTGT is used to obtain tickets to servers that require the client to provide authorization information. This request is represented by arrow **172** in FIG. **4**B. The request includes the TGT that was received earlier from the KDC **109**.

The KDC **109** receives the request from the client **166** and responds to the request by returning a ticket for the AS **107** to the client **166** that includes the digitally signed and sealed certificate **118** (step **156**). The sending of the ticket to the AS **107** is represented by arrow **174** in FIG. **4**B.

The client **166** then requests a PAC from the AS **107**. This PAC will eventually be incorporated into the CTGT that is ultimately issued to the client **166** (step **158**). The request is represented by arrow **176** in FIG. **4**B. The AS **107** normally accesses an authorization database to obtain authorization information. However, in the present instance, since the user is logging on using a logon certificate, the user is outside his home domain, and thus, the authorization database holds no information about the user. Thus, to provide the PAC, the AS **107** uses the domain's public key to decrypt the digital signature and thus, obtains the symmetric key used for encrypting the certificate of credentials and the hash value of its contents. The AS **107** decrypts the certificate of credentials and recomputes the hash of decrypted contents and that must match the hash value obtained from the digital signature. Having verified this and also ascertaining the validity of other aspects of certificate, the AS **107** prepares a PAC from the contents of the digitally signed and sealed certificate **118** in the ticket to craft user's rights and privileges. The AS **107** also marks this PAC as the one generated via logon certificate, a fact that can be used by the local security policy of the target domain. Those skilled in the art will appreciate the fact that the AS **107** can augment the contents of the PAC with additional privileges or restrictions. It then seals the PAC with the secret key that it shares with the KDC before returning it to the client. The PAC is sealed by encrypting it with the secret key of the KDC **109** so that the client cannot access it. The AS **107** returns the sealed PAC (note arrow **178** in FIG. **4**B) to the client **166** (step **160**).

The client **166** sends a message to the KDC **109**, along with the sealed PAC (note arrow **180** in FIG. **4**B) to get a CTGT from the KDC (step **162**). The KDC **109** generates the CTGT so as to include the PAC and forwards (note arrow **182** in FIG. **4**B) the CTGT to the client (step **148**). The CTGT contains the IDs for the user, the privileges of the user, group memberships of the user, and is sealed by encryption using the secret key of the PS. At this juncture, the user appears as if he had an account in the domain to which he is connected. If the user wishes to access a server, he may use his TGT or CTGT.

When a machine is initially booted and is connected to the distributed system **100**, the above steps beginning with step **129** are repeated. The machine takes the place of the user in these steps. Thus, the machine may be validated as a proper machine to be part of the distributed system **100**.

As mentioned above, logon certificates **110** are both revocable and expirable. As to the expirable nature of the logon certificates **110**, each certificate includes a start time at which it becomes valid and an expiration time at which it becomes invalid. In order for a logon certificate **110** to be valid, the start time of the logon certificate must have already been reached and the expiration time of the logon certificate **110** must not yet have been reached.

A logon certificate **110** is revocable and may be explicitly revoked. When an account is inactivated or an administrator instructs the system to revoke logon certificates **110**, the domain controller **106** of the issuing domain circulates a message that lists all user IDs having their logon certificates revoked. This message is signed by the private key of the issuing domain. Other domain controllers **106** utilize this message to determine which parties requesting logon have sufficient credentials. In some instance, it may be necessary for a domain to change its public and private keys and propagate the new public and private key to other domain controllers **106**. As a result, the existing logon certificates **110** issued by the domain are no longer valid.

It should be appreciated that each domain controller **106** can mark certain resources within the domain that are to be accessible to visitors that logon via logon certificates **110**. A special account may be provided within the namespace of the domain to handle such visitors.

While the present invention has been described with reference to a preferred embodiment thereof, those skilled in the art will appreciate that various changes in form and detail may be made without departing from the spirit and the scope of the present invention is defined in the appended claims.

We claim:

1. In a distributed system having computer resources and a facility for checking credentials information to authenticate principals and provide with authorization data, a method for authenticating and authorizing principals comprising the computer implemented steps of:

providing a principal with a secure package holding credentials information for a client;

receiving a principal request to connect to the distributed system at a location in the distributed system that lacks credentials information about the principal to gain access to at least some of the computing resources;

accessing the credentials information held in the secure package to enable the facility for checking credentials information to determine whether the principal is authorized and authenticated to be connected to the distributed system without obtaining credentials information about the principal from a source other than the secure package;

where the principal is not authorized or not authenticated to connect to the distributed system, denying the principal request to be connected to the distributed system; and

where the principal is authorized and authenticated to connect to the distributed system, granting the principal request to be connected to the distributed system.

2. The method of claim **1** wherein the principal is a user.

3. The method of claim **1** wherein the principal is a portable computer.

4. The method of claim **1** wherein the distributed system includes a portable computer having memory and the step of providing the principal with the secure package holding credentials information for the client further comprises a step of loading the secure package holding credentials information for the client into the memory of the portable computer.

5. The method of claim **1** wherein the step of providing the principal with the secure package holding credentials information for the client further comprises a step of storing the secure package on a portable storage medium.

6. The method of claim **5** wherein the portable storage medium is a floppy disk and the step of storing the secure package on the portable storage medium comprises a step of storing the secure package on the floppy disk.

7. The method of claim 1 wherein the step of providing the principal with the secure package holding credentials information for the client comprises a step of providing the principal with an encrypted package holding credentials information for the client.

8. The method of claim 1 wherein the step of providing the principal with the secure package holding credentials information for the client comprises a step of providing a digitally signed and sealed package holding credentials information for a user of the distributed system.

9. The method of claim 1 wherein determining whether the principal is authenticated to be connected to the distributed system comprises the steps of:

(i) determining whether the secure package is authentic; and

(ii) where the secure package is determined to be authentic, determining that the principal is authenticated to be connected to the distributed system.

10. In a distributed system logically partitioned into domains, wherein each user has an associated home domain, a method of maintaining secure access to the distributed system, comprising the computer implemented steps of:

providing a user with a secure package holding credentials information for the user;

receiving a user request to logon to the distributed system in a domain other than the associated home domain of the user, said domain lacking credentials information about the user;

accessing the secure package to examine the credentials information for the user; and

without obtaining credentials information about the user from another source, based on the credentials information for the user provided in the secure package, deciding whether to allow the user to logon or not.

11. The method of claim 10 wherein the distributed system includes a portable computer having memory and the step of providing the user with the secure package holding credentials information for the user comprises loading the secure package holding credentials information for the user into the memory of the portable computer.

12. The method of claim 10 wherein the step of providing the user with the secure package holding credentials information for the user further comprises storing the secure package on a portable storage medium.

13. The method of claim 10 wherein the step of providing the user with the secure package holding credentials information for the user comprises providing the user with an encrypted package holding credentials information for the user.

14. The method of claim 10 wherein the step of providing the user with the secure package holding credentials information for the user comprises providing a digitally signed and sealed package holding credentials information for the user.

15. In a distributed system having a facility for checking credentials information, a method of authorizing connections to the distributed system, comprising the computer implemented steps of:

providing a portable computer with a secure package holding credentials information for the portable computer;

requiring the portable computer to present the secure package when the portable computer wishes to connect to the distributed system at a location lacking credentials information about the portable computer to become part of the distributed system;

examining the credentials information contained within the secure package by the facility for checking credentials information to determine whether the portable computer is authorized to connect to the distributed system,

wherein the portable computer is authorized to connect to the distributed system, allowing the portable computer to connect to the distributed system; and

wherein the portable computer is not authorized to correct to the distributed system, not allowing the portable computer to connect to the distributed system.

16. In a distributed system that is logically partitioned into domains and having a plurality of computers, wherein each computer in the distributed system has an associated home domain, a method of authorizing access to the distributed system, comprising the computer implemented steps of:

providing a secure package at the home domain of a selected computer to the selected computer, said secure package holding credentials information for the selected computer;

receiving a request from the selected computer, to connect to the distributed system at a target domain other than the home domain of the selected computer said target domain lacking credentials information about the selected computer; and

examining the credentials information contained in the secure package to determine whether the selected computer is authorized to be connected to the distributed system at the target domain to become part of the distributed system without obtaining credentials information about the selected computer from a source other than the secure package.

* * * * *

US006678733B1

(12) **United States Patent**
Brown et al.

(10) **Patent No.: US 6,678,733 B1**
(45) **Date of Patent: Jan. 13, 2004**

(54) **METHOD AND SYSTEM FOR AUTHORIZING AND AUTHENTICATING USERS**

(75) Inventors: **Ralph W. Brown**, Boulder, CO (US); **Robert Keller**, Menlo Park, CA (US); **Milo S. Medin**, Sunnyvale, CA (US)

(73) Assignee: **At Home Corporation**, Redwood City, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/428,235**

(22) Filed: **Oct. 26, 1999**

(51) **Int. Cl.**[7] **G06F 15/16**

(52) **U.S. Cl.** **709/229**; 709/217; 709/225; 709/203; 713/172; 713/173; 713/174; 713/159; 725/114; 725/116

(58) **Field of Search** 709/225, 229, 709/217, 203; 713/168, 185, 209, 159, 155, 172-4; 725/91, 93, 114, 116

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,335,346	A		8/1994	Fabbio
5,550,578	A		8/1996	Hoarty et al.
5,586,260	A	*	12/1996	Hu 709/203
5,649,099	A		7/1997	Theimer et al.
5,701,464	A	*	12/1997	Aucsmith 707/10

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

EP	0 748 095 A2	12/1996
EP	0 828 208 A2	3/1998
WO	WO 98/44404 A1	10/1998

OTHER PUBLICATIONS

S.P. Miller et al., "Kerberos Authentication and Authorization System," Oct. 27, 1988 Project Athena Technical Plan pub by Mass Inst. of Technology.*

Paul Chapple, "Rethinking the Role of an Embedded Internet Client In Digital Set-Top Boxes," submitted for World Wide Web Consortium Workshop, "Television and the Web," Jun. 29-30, 1998, Sophia-Antipolis, France.

The World Wide Web Consortium, List of papers prepared for the workshop, "Television and the Web," Jun. 29-30, 1998, Sophia-Antipolis, France.

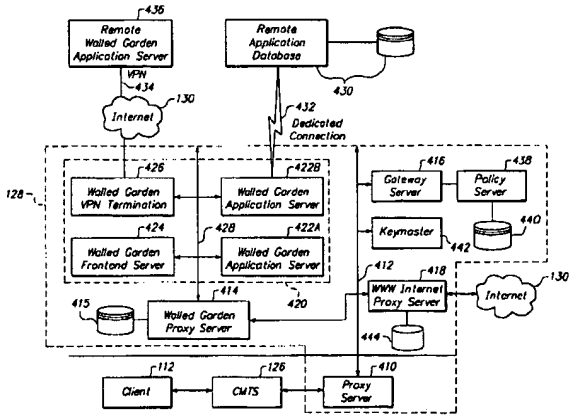
Jupiter Media Metrix, Press Release Mar. 1, 1999. "Jupiter: Web Ventures Are Woefully Unprepared For Set-Top Future." http://www.jmm.com/xp/jmm/press/199_pr_030199b.xml.

Primary Examiner—Krisna Lim
Assistant Examiner—Anita Choudhary
(74) *Attorney, Agent, or Firm*—Fenwick & West LLP

(57) **ABSTRACT**

A walled garden contains links to one or more servers providing network-based services. A walled garden proxy server (WGPS) controls access to the walled garden. When a user of a client wishes to access a service in the walled garden, the client sends a request to the WGPS including a plot number identifying the service and a ticket granting the client access to the service. The WGPS denies access to clients lacking a ticket or presenting invalid tickets. In response, the client contacts a gateway server (GS) having a database of users and associated access rights. The user presents authentication information to the GS. If the user positively authenticates, the GS generates a ticket containing a Box ID from the client, an expiration date, and set of bits representing the access rights of the user. The GS encrypts the ticket and gives it to the client. When the WGPS receives a request to access a service in the walled garden, it decrypts the ticket and uses the plot number as an index into the set of bits representing the user access rights. The indexed value indicates whether the WGPS allows the client to access the service. Accordingly, services provided by the walled garden can be sold individually or in tiers.

25 Claims, 5 Drawing Sheets



U.S. PATENT DOCUMENTS

5,815,574	A *	9/1998	Fortinsky	713/153
5,905,872	A	5/1999	DeSimone et al.	
5,918,013	A	6/1999	Mighdoll et al.	
5,941,947	A	8/1999	Brown et al.	
5,950,195	A	9/1999	Stockwell et al.	
6,003,776	A *	12/1999	Drupsteen	235/492
6,038,319	A	3/2000	Chari	
6,049,877	A	4/2000	White	
6,101,535	A *	8/2000	Hussmann et al.	709/217
6,101,607	A	8/2000	Bachand et al.	713/201
6,119,945	A *	9/2000	Muller et al.	235/492
6,134,551	A *	10/2000	Aucsmith	707/10
6,192,349	B1 *	2/2001	Husemann et al.	705/65
6,216,227	B1 *	4/2001	Goldstein et al.	713/172
6,260,027	B1 *	7/2001	Takahashi et al.	705/69
6,298,482	B1	10/2001	Seidman et al.	
6,311,207	B1 *	10/2001	Mighdoll et al.	709/203
6,321,337	B1	11/2001	Reshef et al.	
6,343,324	B1 *	1/2002	Hubis et al.	709/229
6,351,812	B1 *	2/2002	Datar et al.	713/182
6,505,300	B2 *	1/2003	Chan et al.	709/229
6,515,598	B2 *	2/2003	Parenteau et al.	341/60

* cited by examiner

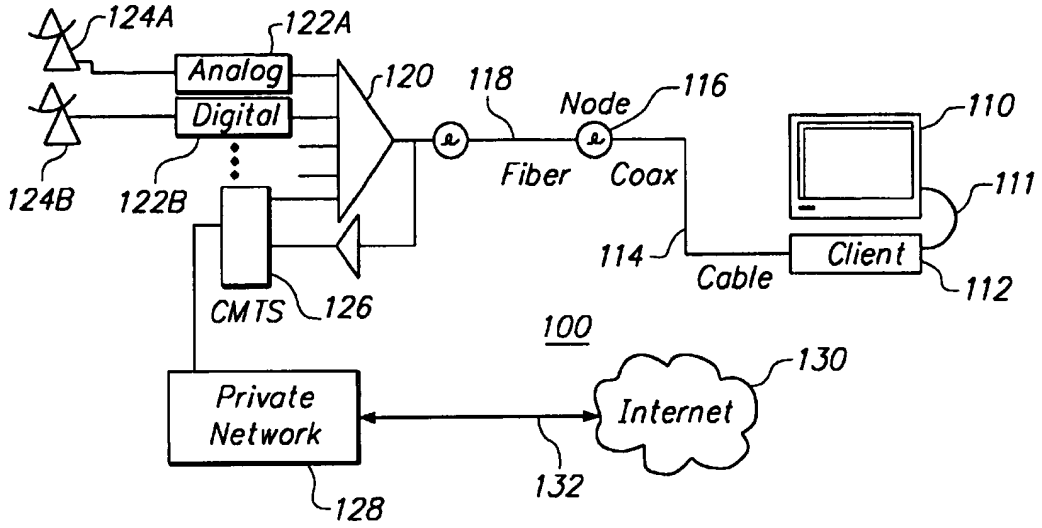

124A
122A
Analog
Digital
120
118
Node
116
110
124B
122B
Fiber
Coax
111
114
Cable
Client
112
CMTS
126
100
130
Private Network
Internet
132
128

FIG. 1

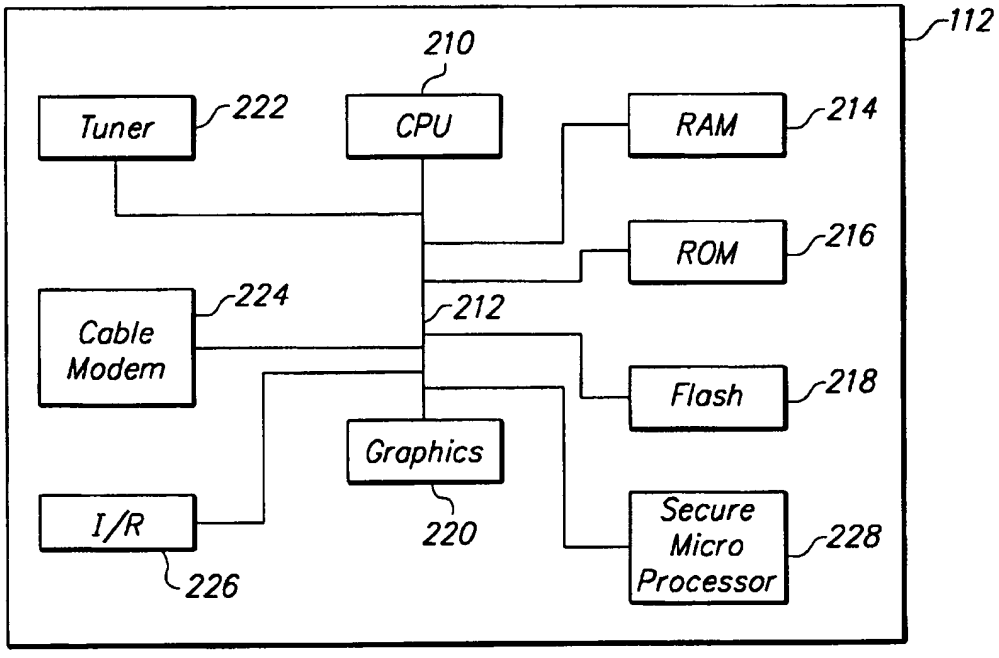

112
210
Tuner
222
CPU
RAM
214
ROM
216
Cable Modem
224
212
Flash
218
Graphics
220
I/R
226
Secure Micro Processor
228

FIG. 2

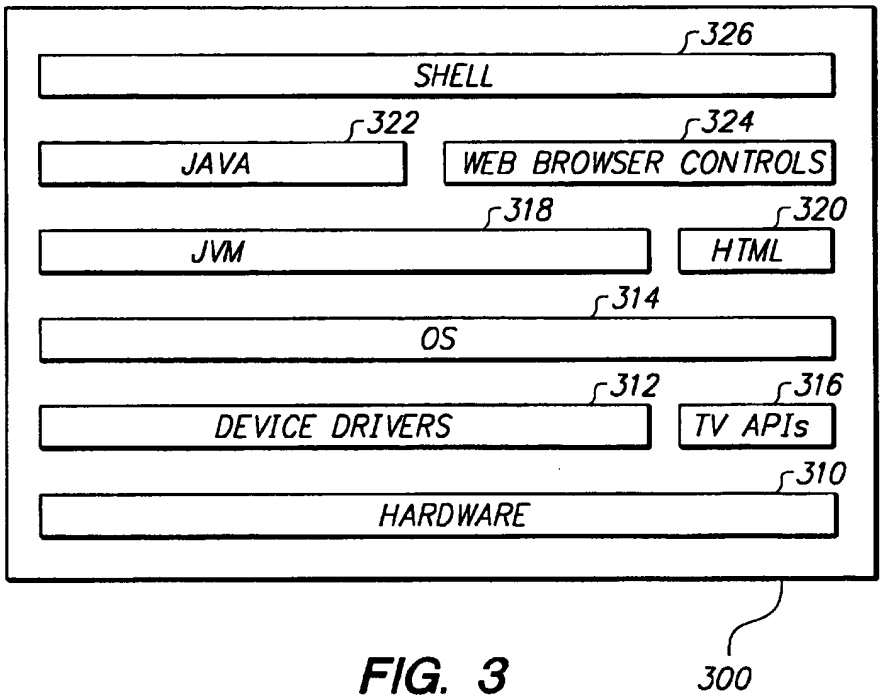

326
SHELL
322
JAVA
324
WEB BROWSER CONTROLS
318
JVM
320
HTML
314
OS
312
DEVICE DRIVERS
316
TV APIs
310
HARDWARE
300

FIG. 3

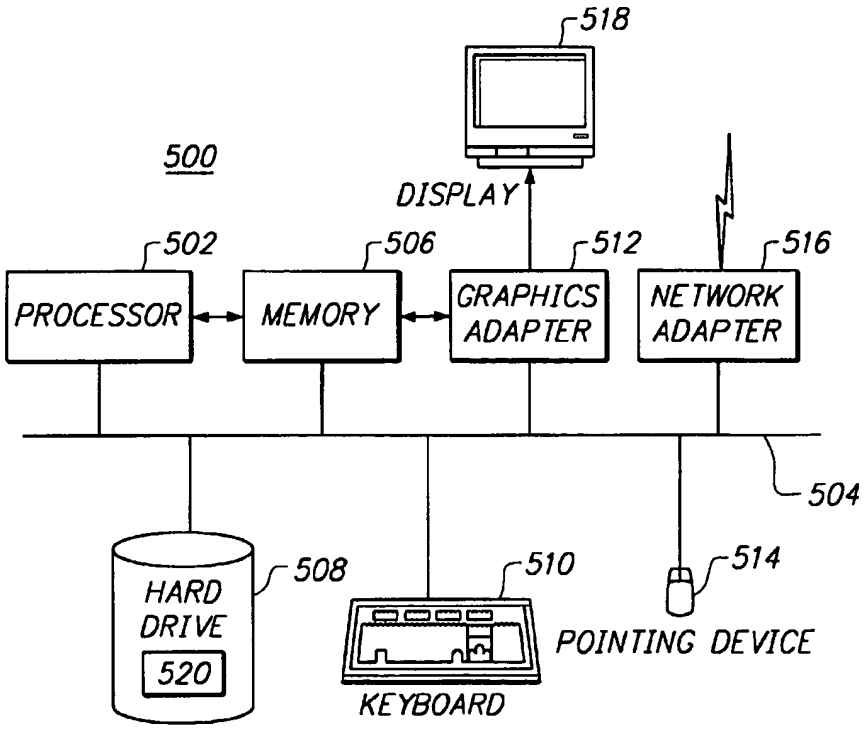

518
500
DISPLAY
502
506
512
516
PROCESSOR
MEMORY
GRAPHICS ADAPTER
NETWORK ADAPTER
504
HARD DRIVE
520
508
510
514
POINTING DEVICE
KEYBOARD

FIG. 5

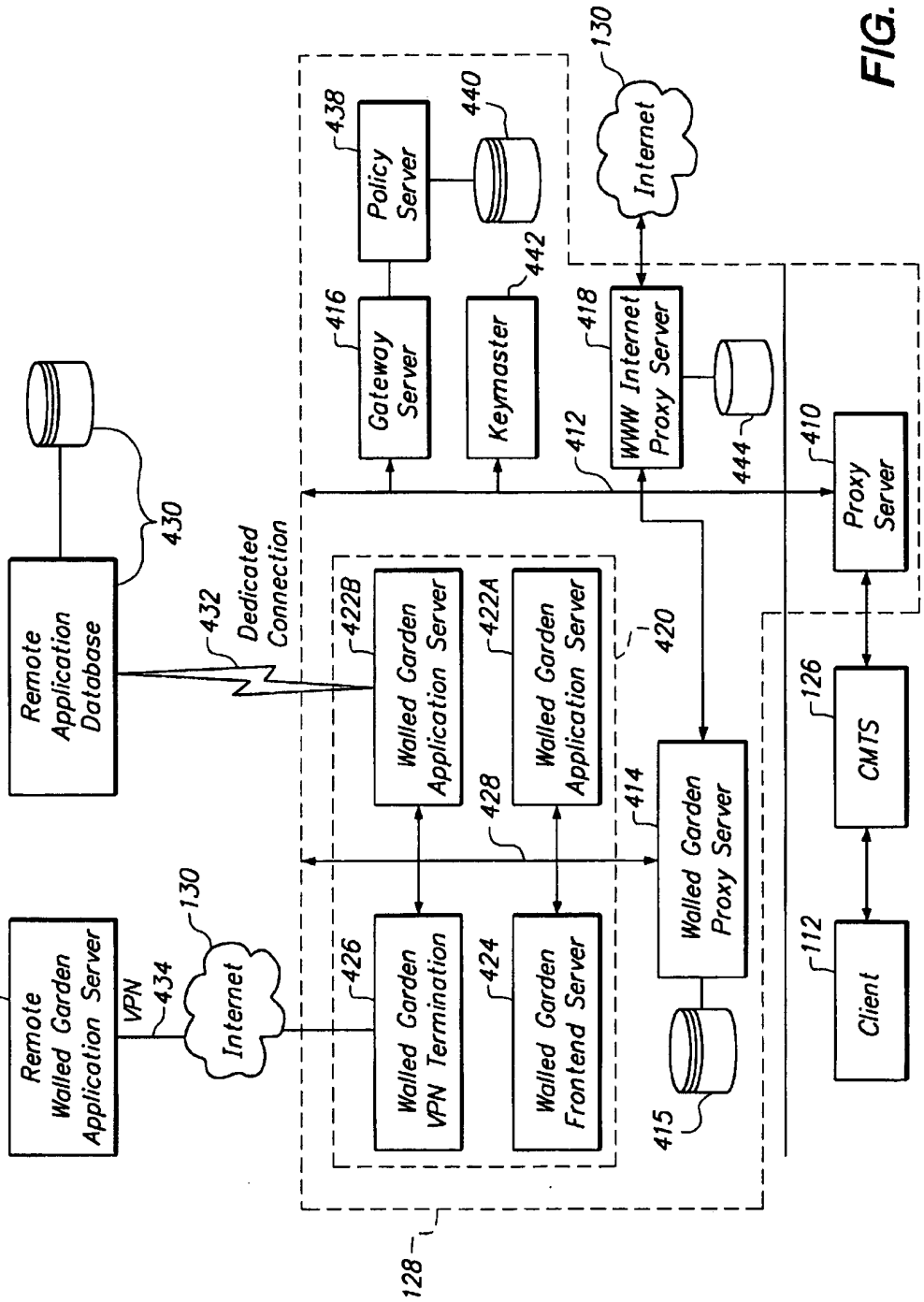

Remote Application Database
430
Dedicated Connection
432
Remote Walled Garden Application Server
436
VPN
434
Internet
130
Walled Garden VPN Termination
426
Walled Garden Frontend Server
424
Walled Garden Application Server
422B
Walled Garden Application Server
422A
420
428
Walled Garden Proxy Server
414
415
128
Gateway Server
416
Policy Server
438
Keymaster
442
440
WWW Internet Proxy Server
418
412
444
Internet
130
Proxy Server
410
CMTS
126
Client
112

FIG. 4

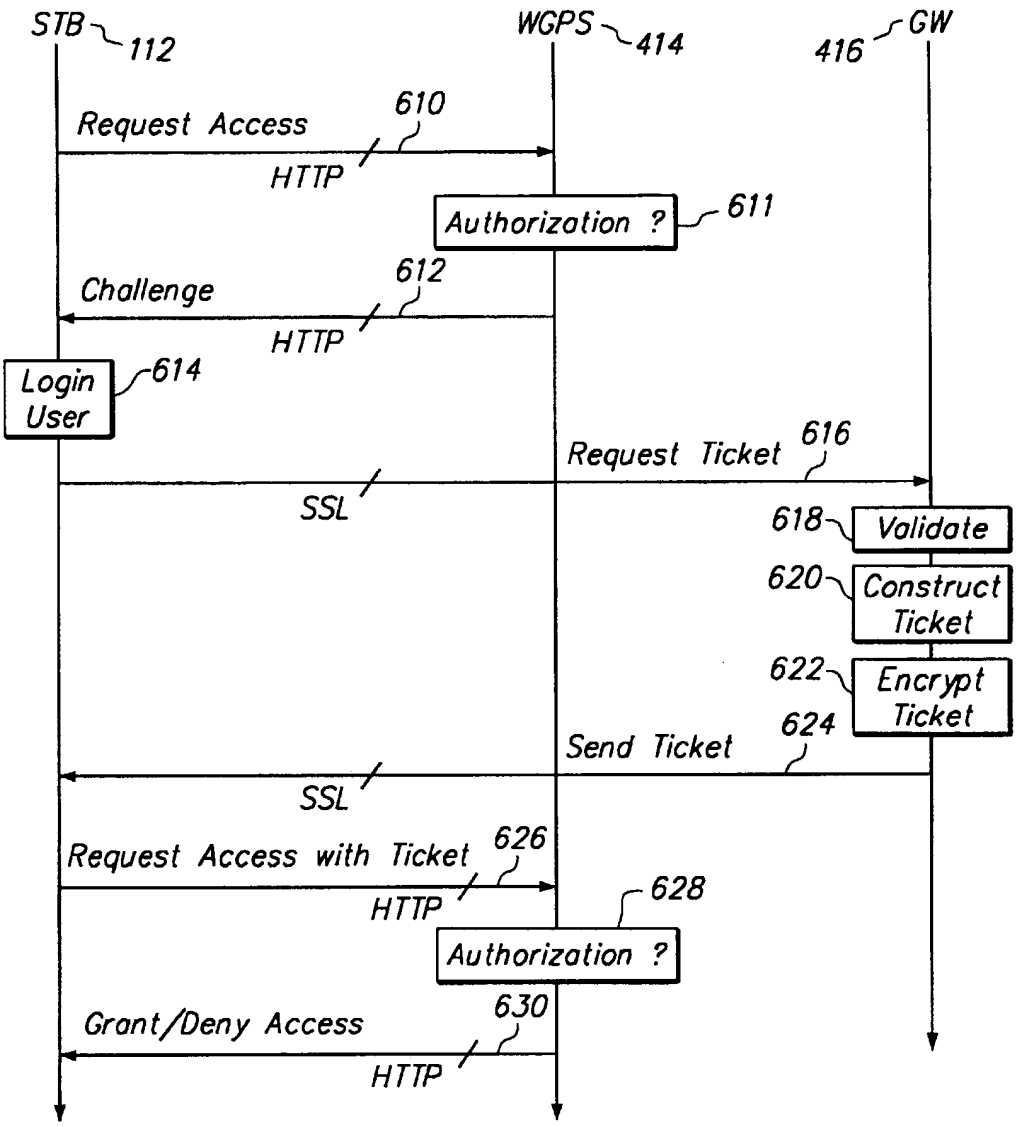

STB
112
WGPS
414
416
GW
Request Access
610
HTTP
Authorization ?
611
Challenge
612
HTTP
Login User
614
Request Ticket
616
SSL
618
Validate
620
Construct Ticket
622
Encrypt Ticket
624
Send Ticket
SSL
Request Access with Ticket
626
HTTP
628
Authorization ?
Grant/Deny Access
630
HTTP

FIG. 6

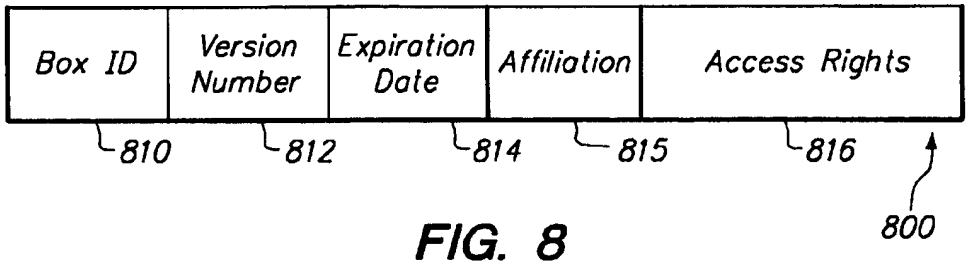

Box ID
Version Number
Expiration Date
Affiliation
Access Rights
810
812
814
815
816
800

FIG. 8

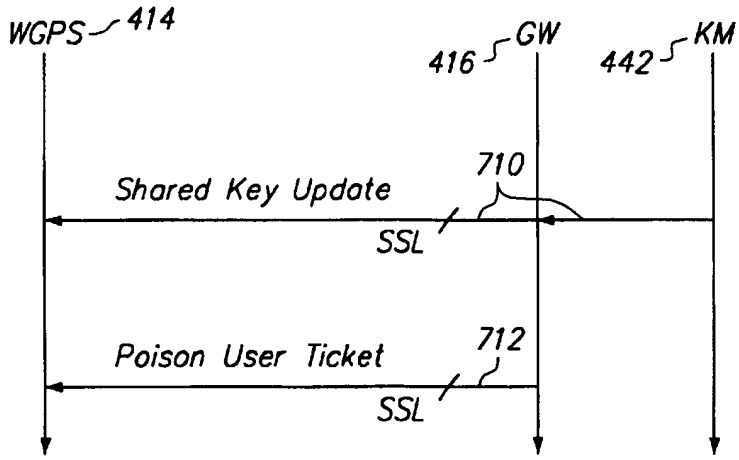

WGPS
414
416
GW
442
KM
710
Shared Key Update
SSL
712
Poison User Ticket
SSL

FIG. 7

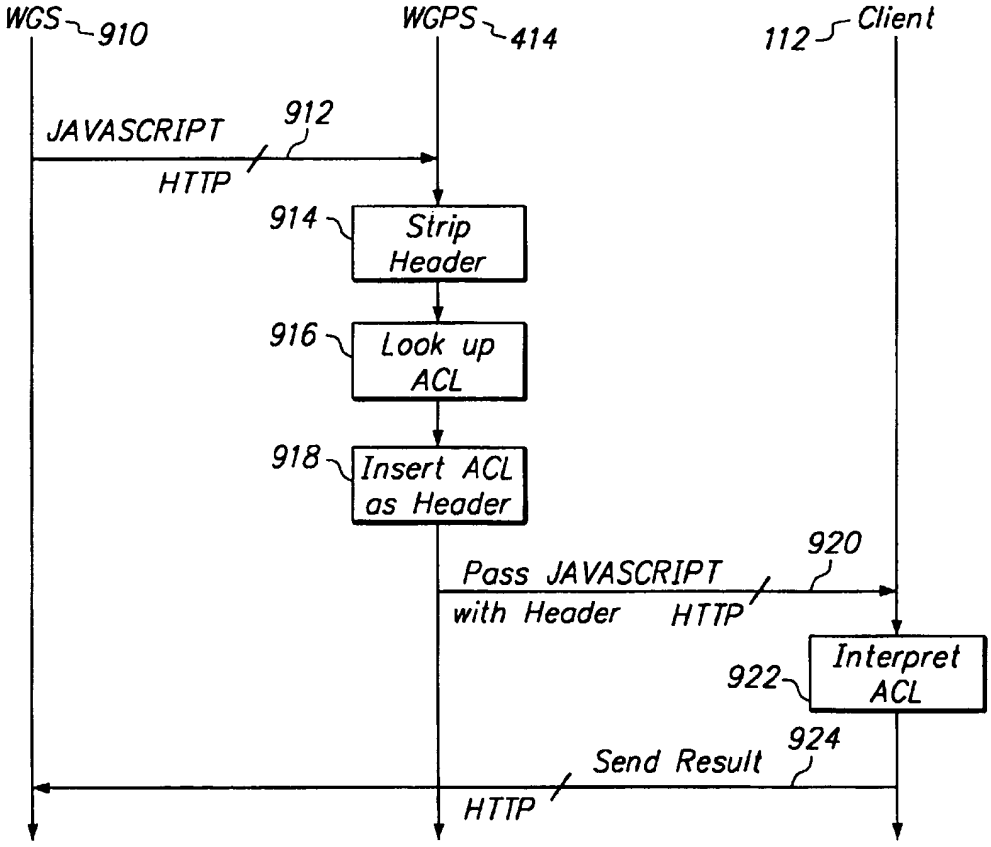

WGS
910
WGPS
414
112
Client
912
JAVASCRIPT
HTTP
914
Strip Header
916
Look up ACL
918
Insert ACL as Header
920
Pass JAVASCRIPT with Header
HTTP
922
Interpret ACL
924
Send Result
HTTP

FIG. 9

METHOD AND SYSTEM FOR AUTHORIZING AND AUTHENTICATING USERS

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is related to U.S. Pat. No. 6,370,571, entitled SYSTEM AND METHOD FOR DELIVERING HIGH-PERFORMANCE ONLINE MULTIMEDIA SERVICES, which issued on Apr. 9, 2002 and is hereby incorporated by reference herein and U.S. patent application Ser. No. 09/427,778, entitled METHOD AND SYSTEM FOR RESTRICTING ACCESS TO USER RESOURCES, filed on even date herewith by Ralph W. Brown, Robert Keller, David Tempkin, and Milo S. Medin, which is hereby incorporated by reference herein.

BACKGROUND

1. Field of the Invention

This invention pertains in general to high-speed data networks and in particular to a system and method for authenticating and authorizing users seeking access to resources on a network.

2. Background of the Invention

Cable television service is usually sold as a subscription to one or more "tiers" of channels. Typically, a basic cable subscription allows access to a small tier of channels and is sold for a relatively low subscription fee. The subscriber can purchase additional tiers of cable channels for additional fees.

In most cable systems, the subscriber uses a "set-top box" (STB) to access the cable channels. The STB contains a microprocessor and other hardware for tuning and descrambling channels in the tiers to which the subscriber has access. The STB may also enable other services, such as pay-per-view or digital music reception in return for additional fees.

In recent years, the STB has incorporated a cable modem that enables access to Internet- and World Wide Web- ("the web") based resources via the cable infrastructure. A cable modem typically has at least one assigned Internet Protocol (IP) address and is managed by an Internet Service Provider (ISP). The ISP inserts and extracts Internet traffic to and from the cable infrastructure and distributes it to the cable modem having the given IP address or the Internet, as appropriate.

In the standard Internet model, any computer on the network can communicate with any other computer. As a result, any cable modem serviced by the ISP can access any server providing a service available on the Internet. This Internet access model is different than the traditional cable television model since there is no easy way to restrict subscribers to only certain servers or resources on the network.

This Internet access model may be acceptable under current usage habits. However, as web-based electronic commerce becomes more commonplace and televisions and the Internet become more tightly coupled, there is an increasing desire to sell network-based services to subscribers. ISP's prefer to sell network-based services using subscriptions analogous to cable television channel subscriptions. Given current technology, however, it is difficult for the ISP to sell tiers of network-based services in the same way that the cable television provider sells tiers of channels.

In addition, there is no easy client-side way to restrict access to network-based services by individual users of a

STB. As far as the ISP and servers on the network are concerned, every transaction to or from the IP address of the STB has the same rights and privileges. Server-side authentication, such as requiring a password before allowing access to a web site, is well known but inconvenient to both the web site operator and the STB user. The web site must manage a list of users and passwords and the user must supply the user name and password at each web site.

Accordingly, there is a need for a way to provide tiers of network-based services to cable modem users. Ideally, the solution to this need will also allow access to be restricted at the level of the individual user.

SUMMARY OF THE INVENTION

The above needs are met by a method and system that authenticates users and authorizes the users to access a walled garden of network services. A user has a client, which is preferably a set top box (STB) coupled to a television set or computer system. The client preferably contains a central processing unit, memory, a television tuner, and a cable modem. The client also preferably contains a video subsystem for generating images on the display and an input for accepting commands from a user.

The client preferably executes software supporting standard web browsing functionality. In one embodiment, the client executes the Windows CE operating system. Programs executing on the operating system include a hypertext markup language (HTML) rendering engine, a JAVA virtual machine for executing JAVA programs, and other controls supporting web browsing functionality. A shell program also preferably executes on the operating system and generates a user interface on the television or computer display coupled to the client.

The cable modem is preferably coupled to a coaxial cable and supports bi-directional broadband communications with a private network using the Internet protocol (IP). The coaxial cable is typically aggregated with other cables into a fiber-optic cable. The fiber-optic cable, in turn, is coupled to a cable modem termination server (CMTS) at a headend. The CMTS contains hardware for terminating the IP data channel, including IP switches, routers, and high-availability servers. The client can also use other broadband communications media, such as digital subscriber line (DSL) and wireless modems to communicate with the private network.

The private network contains a walled garden proxy server (WGPS), a gateway server (GS), and an Internet proxy server. The WGPS controls access to a walled garden providing network-based services. The services available on the walled garden may include, for example, access to electronic content, access to electronic commerce services, and any other functionality that can be provided electronically via a network. These services are provided by one or more walled garden servers coupled to a walled garden network. The walled garden servers may include servers directly coupled to the walled garden network, servers having direct connections to remote application databases, servers coupled to the walled garden network via a virtual private network, and servers having only a frontend on the walled garden network. Each walled garden server is identified by a plot number.

The GS acts as a gateway to a policy server (PS). The PS is coupled to a database of information describing the walled garden access rights of the users. A keymaster provides encryption and decryption keys to the various servers.

When a user wishes to access a service in the walled garden, the client sends a hypertext transport protocol

(HTTP) request to the WGPS identifying the plot number of the requested service. If the client has a ticket granting access to the walled garden, the client includes the ticket in an authorization header. If the client does not provide a ticket or the ticket is invalid, the WGPS denies the HTTP request.

In response to a denial, the client sends a message to the GS requesting a ticket. The user authenticates himself or herself to the client by providing authentication information and the client provides this information to the GS. Assuming the user is authenticated, the GS uses the PS to look up the user in the database and determine the services in the walled garden to which the user has access. Then, the GS constructs a ticket including a bit field indicating the user's access rights, an expiration date, and other information. The PS preferably encrypts the ticket with the encryption key received from the keymaster and transmits the encrypted ticket to the client. The client cannot decrypt or alter the ticket because the client does not know the key.

Then, the client sends the WGPS a new request to access a service in the walled garden and includes the ticket. The WGPS decrypts the ticket using the key and verifies that the ticket is valid. Preferably, the WGPS uses the plot number supplied by the client as an index into the bit field in the ticket. The corresponding bit specifies whether the user has authorization to access the requested walled garden service. If the ticket authorizes the user to access the service, then the WGPS grants the client's HTTP request.

Accordingly, tiers of services can be maintained by placing the services in the walled garden and giving the user access to only those services in the tier to which the user subscribes. The user, in turn, must authenticate itself only once.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram illustrating a high-level view of a network architecture according to a preferred embodiment of the present invention;

FIG. **2** is a high-level block diagram of a client according to an embodiment of the present invention;

FIG. **3** is a block diagram illustrating the various levels of software and hardware executing on the client illustrated in FIG. **2**;

FIG. **4** is a high-level logical block diagram illustrating the client, a private network, the Internet, and other related entities;

FIG. **5** is a high-level block diagram of a computer system for performing as one of the servers illustrated in FIG. **4**, such as the walled garden proxy server (WGPS) and/or the gateway server (GS);

FIG. **6** is a flow diagram illustrating transactions among the client, WGPS, GS, and keymaster according to a preferred embodiment of the present invention;

FIG. **7** is a flow diagram illustrating transactions between the WGPS, GS, and keymaster that may occur independently of the transactions of FIG. **6**;

FIG. **8** illustrates the fields in a ticket according to an embodiment of the present invention; and

FIG. **9** is a flow diagram illustrating transactions among a walled garden site, the WGPS, and the client according to a preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

FIG. **1** is a block diagram illustrating a high-level view of the network architecture **100** according to a preferred embodiment of the present invention. A television (TV) **110** is coupled via a data link **111** to a client **112**, which is preferably a set top box (STB). The TV **110** can be any form of television, including a conventional analog TV or a high-definition (HD) digital TV. In addition, the TV **110** may be a monitor coupled to the client **112** directly. Alternatively, the client **112** may be coupled to a computer system instead of a TV **110**. The data link **111** may be, for example, a serial link, a radio frequency (RF) link, a broadband link, a standard monitor cable, or an S-video link. The client **112** can also be integrated into the TV or computer system.

Typically, the client **112** is an STB and is purchased or leased by a person or household who also subscribes to the cable TV and data communication services. The person or persons who performs this task is often referred to as the "subscriber." Because multiple people may use a single client **112**, the person using the client in this description is referred to as a "user." The STB **112** or other device that the user uses to access data communication services is generically referred to as a "client." The distinction between actions performed by the user and client are often blurred, especially when the client performs an action, such as fetching a web page, on behalf of a user. Accordingly, the terms "user" and "client" are often interchangeable in this description.

The client **112** preferably includes at least one tuner for tuning a TV signal and a cable modem for receiving and transmitting data. In addition, the client **112** is preferably coupled to a coaxial cable **114** carrying analog and/or digital TV signals and providing two-way data communication between the client **112** and the rest of the network using the Internet protocol (IP). In alternative embodiments, data communication between the client **112** and the rest of the network may be provided by other forms of communication technologies, including analog modem, digital subscriber line (DSL), and wireless technologies. These alternative technologies may or may not carry TV or other video signals with the data.

In the embodiment where the client **112** is coupled to a coaxial cable, the coaxial cable **114** is aggregated with other cables at a node **116**, typically from geographically proximate locations, into a fiber-optic cable **118**. The fiber-optic cable **118**, in turn, is aggregated with other fiber-optic cables at a headend **120**.

The headend **120** integrates signals from multiple sources into the fiber-optic cable **118**. In one embodiment, the headend **120** receives analog **122**A and digital **122**B television signals via analog **124**A and digital **124**B satellite downlinks, respectively. In addition, the headend **120** includes a cable modem termination server (CMTS) **126** for terminating the IP data channel, including IP switches, routers, and high-availability servers.

The CMTS **126** is preferably coupled to a private network **128** maintained by an Internet service provider (ISP) or other organization. In a preferred embodiment of the present invention, the private network **128** includes the high-speed network architecture described in U.S. Pat. No. 6,370,571, entitled SYSTEM AND METHOD FOR DELIVERING HIGH-PERFORMANCE ONLINE MULTIMEDIA SERVICES, which issued on Apr. 9, 2002 and is hereby incorporated by reference herein. In general, the private network **128** provides network access to the users by managing a cable modem contained within the client **112**. A widely accepted standard for cable modems is the Multimedia Cable Network System (MCNS) Data-Over-Cable Service Interface Specifications (DOCSIS). The private net-

work **128** also provides connectivity to servers providing services to the clients, such as caching, account and billing management, electronic commerce, information databases, and any other functionality that can be achieved via a network. Typically, the resources on the private network **128** can be accessed by only subscribers of the ISP.

In the illustrated embodiment, the private network **128** is in communication with the Internet **130** via a network link **132**. For security purposes, a preferred embodiment of the present invention restricts the ability of the client **112** to download software from the Internet **130**. However, other embodiments may provide the client **112** with full access to the Internet **130** or restrict the client to only the resources available in the private network **128**.

FIG. **2** is a high-level block diagram of a client **112** according to an embodiment of the present invention. Preferably, the client **112** is an STB and supports both broadcast services including video, audio, and data, and IP-based services including e-mail, web browsing, IP multicast, and streaming video and audio. In a preferred embodiment of the present invention, the client **112** is a GENERAL INSTRUMENTS DCT-5000, although any STB, computer system, or modem providing equivalent functionality can be substituted. The client **112** preferably includes components typically found in a computer system including a central processing unit (CPU) **210**. The CPU **210** can be any type of general or specific purpose processor and processes instructions and data for providing the functionality described herein. The CPU **210** is coupled via a bus **212** to a random access memory (RAM) **214**, a read only memory (ROM) **216**, and non-volatile memory **218**, such as flash memory. The RAM **214** is preferably synchronous dynamic RAM (SDRAM) and stores programs and data used by the CPU **210** as is well known in the art. Similarly, the ROM **216** and flash memory **218** store instructions and data used to maintain the system operation and configuration.

The bus **212** also couples the CPU **210** with a graphics and video subsystem **220** which generates text and video images on the TV **110**. In addition to providing conventional TV images, the graphics and video subsystem **220** preferably generates a user interface (UI) by which the user can access the features and services provided by the client **112**. The graphics and video subsystem **220** may also support advanced features such as 3-D graphics, video decoding, and video digitizing.

TV and cable modem tuners **222, 224** are also preferably coupled to the bus **212**. The TV tuner **222** is preferably a frequency-agile tuner dedicated to analog and digital broadcast services. The cable modem tuner **224** is preferably a frequency-agile tuner dedicated to cable modem services. Although not shown in FIG. **2**, the client **112** may also include other tuners for handling an out-of-band data channel and a return signal path. The tuners **222, 224** receive the signal from the coaxial cable **114**.

An infrared (IR) transceiver **226** is also preferably coupled to the bus **212**. The transceiver **226** can communicate with a wireless remote control or keyboard, thereby allowing a user to access the features of the client **112** either directly or via on-screen menus. The client **112** also preferably includes a secure microprocessor **228** for supporting secure transactions as described below. The secure microprocessor **228** holds a unique identification value for the client **112** called the "Box ID," a private/public key pair, and other information that can be used to authorize and authenticate messages to and from the client **112**. In alternative embodiments, the client **112** may also include an audio accelerator for performing audio processing, an input/output (I/O) controller for communicating with external devices such as storage devices and hard copy output devices, and/or a network adapter for communicating with a local-area network.

FIG. **3** is a block diagram illustrating the various levels of client software program modules and hardware executing on the client **112** illustrated in FIG. **2**. At the lowest level is the hardware **310** for executing the software and performing the basic functions supported by the client **112**. Device drivers **312** for the various hardware elements sit between the operating system (OS) **314** and the hardware **310**. In one embodiment, the OS **314** is a version of the WINDOWS CE operating system from MICROSOFT CORPORATION of Redmond, Wash. However, other OS's may also be used. The OS **314** controls the operation of the client **112** and makes the features of the client available to other programs executing on the client **112**. The OS **314** uses the device drivers **312** to communicate with the hardware **310**. In addition, a TV application programming interface (API) **316** also sits between the OS **314** and the hardware **310**. The OS **313** uses the TV API **316** to access dedicated TV functionality within the hardware **310**.

A JAVA virtual machine (JVM) **318** and hypertext markup language (HTML) rendering engine **320** preferably execute on the OS **314**. The WM **318** functions as a virtual machine and provides an execution space for JAVA programs **322**. The JAVA programs **322** may be stored locally on the client **112** or downloaded from the private network **128** or the Internet **130**. In addition, the JAVA programs **322** may utilize JAVA classes dedicated to supporting the TV and media functions available on the client **112**. Similarly, the HTML rendering engine **320** supports traditional web browsing functionality. A user can use the web browser controls **324** to navigate through hypertext documents as is well known in the art.

In a preferred embodiment of the present invention, a shell program **326** executes at the highest level. The shell program **326** may be implemented using, for example, native code, JAVA, JAVSCRIPT, ActiveX controls, HTML, and/or dynamic link libraries (DLLs). The shell program **326** is the controlling application and provides the user interface (UI) on the TV **110** and application support for channel navigation, an electronic program guide, storing user preferences, email, and walled garden **420** access.

Preferably, the shell program **326** contains a set of foundation layer APIs **328** that can be called by programs downloaded via the private network **128**. In one embodiment, the functions in the APIs are accessed by JAVASCRIPT code downloaded to the client **112** via HTTP. All functions available through the APIs are subject to access control and a program making use of the APIs must be authorized to access the functions. If a program calls a function for which it is not authorized, the client **112** returns a FAIL_FUNCTION_NOT_AUTHORIZED error status message to the program. This status message indicates to the program that the server that supplied the program is not authorized to perform that function on the client **112**.

Exemplary sets of APIs are described in the Appendix. As described therein, the APIs allow a program to change the television channel to which the client **112** is tuned, inquire about the details of a channel line-up, access an electronic program guide (EPG) stored by the client, instantiate. UI elements on the television **110**, retrieve information about viewer (i.e., user) accounts, access electronic wallet

(E-wallet) functionality in the client to conduct electronic commerce transactions, set reminders for display on the television **110**, and print pages on a printer (not shown) coupled to the client. Additional APIs may allow controlling scaling of the broadcast video picture on the television **110** and accessing settings stored by the client **112**, including user preferences, bookmarks, parental controls, and diagnostics. Other APIs can easily be added to the shell **326** to provide functionality desired by the ISP, server, or users. Preferably, each function in the APIs is named, numbered, or otherwise uniquely identified. Likewise, groups of functions, related or otherwise, may also be named, numbered, or otherwise identified.

FIG. **4** is a high-level logical block diagram illustrating the client **112**, the private network **128**, the Internet **130**, and other related entities. As shown in FIG. **4**, the CMTS **126** is preferably coupled to a proxy server **410**. The proxy server **410** provides caching of web pages and other data for the clients served by the CMTS **126**. The proxy server **410** is connected to a network backbone **412**. A walled garden proxy server (WGPS) **414**, gateway server (GS) **416**, and WWW Internet proxy server **418** (Internet server) are also coupled to the network backbone **412**.

Preferably, the client **112** communicates with the servers on the network **412** using standard communications protocols including the IP, hypertext transport protocol (HTTP), and secure sockets layer (SSL). Communications between the client **112** and the various servers often takes the form of hypertext markup language (HTML) documents, extensible markup language (XML) documents, JAVASCRIPT programs, and data provided through forms. Servers and data on the network **412** are preferably identified with uniform resource locators (URLs).

Each user of the client **112** preferably has a unique identification. A user can log into the client **112** by inputting the user's identity and a personal identification number (PIN) or other form of password. This user information is preferably stored in a local database held in, for example, the non-volatile memory **218** or a storage device. The database has a record for each user of the client **112** and associates the record with the user's login information. The client **112** can provide the user's login information to other servers in the network **128** when necessary to authenticate the user. For security, the user records stored in the client **112** are opaque and cannot be viewed without the login information of the particular user. When a user logs into the client **112**, the login preferably remains valid until the user explicitly logs out or the client **112** is turned off. If no user has logged into the client **112**, one embodiment of the present invention uses a default user profile. The rights and privileges of the default user profile can be set by the ISP.

The WGPS **414** is the entry point for the walled garden **420**. Although FIG. **4** illustrates only a single walled garden **420**, an embodiment of the present invention can have multiple walled gardens controlled by a single WGPS or by multiple WGPS'. Each walled garden may be controlled by a different multiple systems operator (MSO) (e.g., a different cable television company).

The illustrated walled garden **420** includes one or more servers which, in turn, hold one or more sites for providing network-based services to the users. The services may include, for example, access to electronic content such as channel guides, magazines, newspapers, stock prices, music, or video, access to electronic commerce services such as stock trading and buying and selling goods, access to a time-based or metered service, and any other functionality that can be provided electronically via a network. Preferably, the services are implemented using a combination of JAVA, XML, HTML, and/or JAVASCRIPT. The servers may be maintained by the MSO, ISP, or by other organizations who have formed business relationships with the party managing the walled garden **420**. In one embodiment, the services in the walled garden **420** are arranged into sets of tiers. Preferably, the user can subscribe to one or more of the services in the walled garden **420** either individually or as part of a tier.

The WGPS **414** has an associated database **415** for holding permissions available to the user and the walled garden sites. To access the walled garden **420**, the client must present a "ticket" to the WGPS **414** specifying the walled garden **420** and services to which the user has access. Alternatively, the ticket may specify only those services which the user does not have access. The database **415** identifies "poisoned" tickets, i.e., those tickets that are no longer accepted and holds keys for decrypting encrypted tickets. The database also holds information identifying the MSO or MSOs who's customers have access to the walled garden **420** in order to ensure that the ticket is affiliated with the particular walled garden. The WGPS **414** uses the ticket and the information in the database **415** to authenticate the user and authorize the user to access the services in the walled garden **420**.

The database **415** also identifies the rights of walled garden sites to access the APIs in the clients **112**. Preferably, the database **415** stores a Walled Garden Permissions Table that specifies the API access rights of each server or site in the walled garden. In one embodiment of the present invention, the permissions table is as follows:

Walled Garden Permissions Table

URL Prefix	User Agent	WG ACL	Affiliation
http:// . . .	DCT-5000	010111	0110
http:// . . .	DCT-5000	010101	0100
http:// . . .	DCT-5000	011101	0001
. . .	. . .	. . .	. . .
http:// . . .	DCT-5000	110101	1000

The permissions table is preferably indexed by URL prefix. The URL Prefix field preferably holds a URL string long enough to uniquely identify the walled garden site having the associated permissions. For example, the URLs "http://disney.com/company/index.html" and "http://disney.com/company/about/index.html" will both match a table entry with the URL prefix "http://disney.com/company/." This technique allows different permissions to be assigned to different subtrees of a site's content.

The User Agent field preferably holds a string identifying the type of browser used by the user. For example, the User Agent field may indicate that the user is using a DCT-5000 STB. Alternatively, the field may indicate that the user is using NETSCAPE NAVIGATOR, MICROSOFT INTERNET EXPLORER, or any other type of browser. Since different user agents may have different API sets and capabilities, sites in the walled garden may have separate permissions table entries for each type of user agent. The client **112** identifies the user agent when it sends a HTTP request to the WGPS **414**.

The Walled Garden Access Control List (WG ACL) field preferably contains a bit-map, or ACL, indicating to which client APIs the walled garden sites having the given URL prefix can access. The mapping from bit position to API

function is arbitrary and extensible. A value of zero preferably means the site does not have permission to invoke the corresponding API function or functions, and a value of one preferably means the site does have permission to invoke the corresponding API function or functions. The Affiliation field identifies the particular walled garden **420** or MSO to which the ACL pertains.

The exemplary walled garden **420** illustrated in FIG. **4** contains two walled garden application servers (WGAS's) **422**A–B, a walled garden front end server (WGFS) **424**, and a walled garden virtual private network (VPN) termination point **426** (WGVPNTP) interconnected by a walled garden network (WGN) **428**. The WGN **428** is preferably a closed network and essentially performs as a local area network coupling the various types of walled garden servers with the WGPS **414**. A WGAS **422** is preferably a web server or some other form of server that provides web-like functionality to a user. A WGAS **422**A may contain all of the hardware and storage necessary to provide a service to a user. Alternatively, the WGAS **422**B may be coupled to a remote application database **430** via a dedicated network connection **432**. This latter embodiment may be preferred, for example, when the WGAS **422**B acts as an interface to a large database of information and the entity managing the WGAS **422**B does not wish to replicate the contents of the database within the walled garden **420**.

The WGFS **424** provides a frontend interface for backend servers located elsewhere on the Internet **130** or otherwise in communication with WGFS **424**. For example, a WGFS **424** may be used when a large organization wishing to have a presence in the walled garden **420** leases server space from the ISP or other entity managing the walled garden. The WGFS **424** provides an access point in the walled garden **420** through which the clients can access the backend servers.

The WGVPNTP **426** allows an organization to maintain a presence in the walled garden **420** using remote servers. The ISP or other entity managing the walled garden **420** establishes a VPN **434** over the Internet **130** connecting the WGVPNTP **426** with a remote WGAS **436**. The remote WGAS **436** communicates through the WGVPNTP **426** to perform the same functions as a local WGAS **422**.

Each unique service within the walled garden **420** is preferably identified by a unique "plot number." The client **112** preferably identifies a specific walled garden service with the URL "http://wg/<plot_number>/ . . . " The plot number is preferably used as an index into the ticket and identifies a value specifying whether the user has access to the service. A walled garden service is typically implemented on a single server. However, a single server can support multiple walled garden services. Accordingly, a server may be identified by more than one plot number, with each plot number mapping to a different site residing on the server. A single service can also reside on multiple servers, such as when load balancing is being employed. In this case, a single plot number may resolve to more than one server.

The GS **416** controls access to a policy server (PS) **438**. The GS **416** preferably receives communications from the client **112** in the form of XML and/or forms via HTTP over SSL and translates the communications into database transactions using protocols such as lightweight directory access protocol (LDAP), SQL, and open database connectivity (ODBC). The GS **416** passes the transactions to the PS **438** and the PS **438** accesses a database **440** of user authorization and authentication information in response. The database contains a list of users, walled gardens, and services in particular walled gardens **420** available to the users. The database **440** does not need to be centralized and, in one embodiment, is distributed on a regional basis. The GS **416** communicates with the PS **438** to authenticate a user's identity and issue the client a ticket specifying the walled gardens and services that the user can access. The GS **416** preferably encrypts the ticket using a secret key shared with the WGPS **424** in order to limit potential attacks on the ticket by the user. The user's client **112** stores the ticket and presents it to the WGPS **414** when seeking to access a walled garden **420**.

The Internet server **418** is essentially the same as the WGPS **414**, except that the Internet server **418** controls access to the Internet **130** at large rather than to the walled garden **420**. In a preferred embodiment, the Internet server **418** has a database **444** for holding permissions indicating web sites that users can access and client API functions that the web sites can access. A client accesses the Internet **130** by presenting a ticket to the Internet server **418** specifying the Internet sites to which the user has access. In one embodiment, the ticket specifies the URLs using regular expression pattern matching. The database **444** also identifies poisoned tickets.

The keymaster **442** provides encryption keys to the GS **416**, WGPS **414**, and Internet Server **418**. Preferably, the keymaster **442** has SSL links, or some other form of secure communication links, to the servers **414**, **416**, **418**. The keymaster **442** generates pseudo-random encryption keys and securely passes the keys to the servers **414**, **416**, **418**. The servers **414**, **416**, **418** use the keys to encrypt and decrypt the tickets. In a preferred embodiment, the servers **414**, **416**, **418** use symmetric encryption and use the same key to encrypt and decrypt tickets, although other encryption systems can be used. Each key is valid for a predetermined time period. The keymaster **442** issues a new key to the servers **414**, **416**, **418** at the expiration of the previous key. Each key is preferably indexed so that the keys can be individually identified.

The entities illustrated in FIG. **4** are not necessarily physically separate or executing on dedicated computer systems. For example, the walled garden network and the network connecting the various servers may actually be a single network that is logically divided into separate networks. Moreover, the various servers, such as the WGPS **414**, GS **416**, and PS **438**, may actually be integrated into the proxy server **410** or another computer system. Likewise, the various databases **415**, **440** may be implemented on a single database. Conversely, the functionality ascribed to a single network, server, or database in the description of FIG. **4** may actually be performed by multiple networks, servers, and/or databases, respectively. Thus, FIG. **4** illustrates logical entities and logical interconnections between the entities according to a preferred embodiment of the present invention. However, alternative embodiments of the. present invention may have different physical structures.

FIG. **5** is a high-level block diagram of a computer system **500** for performing as one or more of the servers, such as the WGPS **414** and/or the PS **428**, illustrated in FIG. **4**. Illustrated are at least one processor **502** coupled to a bus **504**. Also coupled to the bus **504** are a memory **506**, a storage device **508**, a keyboard **510**, a graphics adapter **512**, a pointing device **514**, and a network adapter **516**. A display **518** is coupled to the graphics adapter **512**.

The at least one processor **502** may be any general-purpose processor such as an INTEL x86 compatible- or SUN MICROSYSTEMS SPARC compatible-central processing unit (CPU). The storage device **508** may be any device capable of holding large amounts of data, like a hard

drive, compact disk read-only memory (CD-ROM), DVD, or some form of removable storage device. The memory **506** holds instructions and data used by the processor **502**. The pointing device **514** may be a mouse, track ball, light pen, touch-sensitive display, or other type of pointing device and is used in combination with the keyboard **510** to input data into the computer system **500**. The graphics adapter **512** displays images and other information on the display **518**. The network adapter **516** couples the computer system **500** to a local or wide area network.

Program modules **520** for performing the functionality of the server, according to one embodiment of the present invention, are stored on the storage device **508**, loaded into the memory **506**, and executed by the processor **502**. Alternatively, hardware or software modules may be stored elsewhere within the computer system **500**. In one embodiment of the present invention, one or more of the illustrated servers are implemented using redundant hardware to create a high-availability computer system. As is known in the art, an advantage of a high-availability computer system is a reduced risk of system failure.

FIG. **6** is a flow diagram illustrating transactions among the client **112**, WGPS **414**, GS **416**, and keymaster **442** according to a preferred embodiment of the present invention. The illustrated transaction sequence represents only one of many possible sequences of transactions. In FIG. **6**, a horizontal arrow represents a transaction where the primary flow of information is in the direction of the arrow. The slash across the transaction illustrates the protocol used to transmit the data, typically either HTTP or the SSL. In addition, FIG. **7** is a flow diagram illustrating transactions between the WGPS **414**, GS **416**, and keymaster **442** that may occur independently of the transactions of FIG. **6**.

Initially, the user uses the UI on the client **112** to request **610** access to a service in the walled garden **420**. For example, the client **112** may generate a UI on the TV **110**. The user, using the UI and an input device such as an IR keyboard, requests access to the service through the web browsing software **324** executing on the client **112**. Alternatively, the client **112** may be coupled to or integrated into a computer system and the user may use web browsing software to request access to a web site in the walled garden **420**. As mentioned above, the request **610** from the client **112** to the WGPS **414** preferably takes the form of a URL such as "http://wg/<plot number>/. . . " In one embodiment, the user visits a web page or portal that references, either directly or indirectly, all of the available walled garden services. When the user selects a link to a particular service, the web page directs the client **112** to the proper URL.

The WGPS **414** receives the request **610** and determines from the URL that the client is attempting to access a restricted service in the walled garden **420**. Assume, however, that this request **610** is the first request from the client **112** to the WGPS **414**. As a result, the client **112** did not include a ticket with the request **610**. Therefore, the WGPS **414** denies **611** access to the walled garden **420** and sends a HTTP **407** response to challenge **612** the client **112** to supply the ticket in a subsequent request.

The client **112** receives the challenge **612**. Preferably, the web browser then passes control to an authorization dynamic link library (DLL) executing on the client **112**. The authorization DLL creates the appropriate UI to let the user authenticate himself or herself to the client **112**.

The authorization DLL then establishes a SSL connection with the GS **416** and makes a request **616** for the ticket by sending the user authentication information, as well as the Box ID of the client **112**, across the SSL connection. The GS **416** authenticates the user by validating **618** the authentication information against the information in the database **440**.

If the validation **618** is successful, the GS **416** preferably constructs **620** the ticket. As shown in FIG. **8**, the ticket **800** preferably includes the Box ID **810** of the client **112** requesting the ticket, a version number **812**, an expiration date **814** (or duration when the ticket is valid), an affiliation **815**, and a set of bits representing the access rights of the user **816**. The version number **812** is preferably a control number used by the GS **416** to ensure that the WGPS **414** properly interprets the ticket **800**. The expiration date **814** can be any time in the future or a time span when the ticket **800** is valid and may range, for example, from a few minutes to a few hours. The affiliation indicates the particular walled garden **420** or MSO to which the ticket **800** pertains. The set of bits representing the access rights of the user **816** are preferably organized such that certain bits correspond to certain servers, sites, or services within the walled garden **420**. In one embodiment of the present invention, the bits representing the access rights **816** are run length encoded (RLE) to reduce the storage size of the field. Other information, such as the IP address of the client **112** and a timestamp may also be stored in the ticket **800**.

As shown in FIG. **7**, the keymaster **442** occasionally shares **710** a secret key with the GS **416** and the WGPS **414** via an SSL connection. Returning to FIG. **6**, the GS **416** preferably uses a symmetric encryption technique to encrypt **622** the ticket **800**, T, with the shared secret key to produce an encrypted ticket, T'. In an alternative embodiment, the GS **416** encrypts only the portion of the ticket containing the bits representing the user access rights **816**. The WGPS **414** uses the decryption key, which is preferably identical to the encryption key, to decrypt the ticket **800**. In one embodiment of the present invention, this encryption is performed using the data encryption standard (DES). However, other embodiments of the present invention may use different encryption techniques, including techniques where the encryption and decryption keys differ.

The resulting encrypted ticket is passed **624** to the client **112**. The client **112** preferably stores the encrypted ticket internally. Since the client **112** does not have access to the secret key shared by the keymaster **442**, GS **416**, and WGPS **414**, the client cannot decrypt or alter the ticket.

If, for any reason, the GS **416** decides to invalidate or revoke a ticket, the GS **416** poisons the ticket by sending **712** an invalidity notice to the WGPS **414** as shown in FIG. **7**. The WGPS **414** treats a request to access the walled garden **420** made by a client with a poisoned ticket as if no ticket had been included.

Returning to FIG. **6**, the client **112** again sends a HTTP request to the WGPS **414** requesting access to a service in the walled garden **420**. This time, however, the client **112** includes the encrypted ticket with the HTTP request in an "Authorization" header. The WGPS **414** receives the ticket with the request and determines **628** whether the ticket grants the user access to the walled garden **420** and walled garden service. To make this determination, the WGPS **414** uses the timestamp to determine the secret key used to encrypt the ticket. Then, the WGPS **414** uses the secret key to decrypt the ticket. Next, the WGPS **414** compares the Box ID in the ticket with the Box ID of the requesting client to ensure that the ticket was received from the correct client **112**. The WGPS **414** also checks the expiration date in the ticket to ensure that the ticket is still valid and compares the ticket with the information in its database **415** to ensure that the ticket has not been poisoned.

If the above tests are satisfied, then the WGPS **414** examines the affiliation **815** and the set of bits representing

the access rights of the user **816** to determine whether the user has rights to the specified walled garden **420** service. To make the latter determination, the WGPS **414** extracts the plot number from the HTTP request and uses it as an index into the set of bits **816** in the ticket **800**. Preferably, the value of the indexed bit specifies whether the user is authorized to access the walled garden **420** service or site having the given plot number. This embodiment is preferred because it minimizes the overhead utilized to determine whether the ticket allows access. Of course, alternative embodiments of the present invention may use different techniques to encode the user access rights in the ticket.

The WGPS **414** then either grants or denies **630** access to the user. If the WGPS **414** grants access, then it allows the user request **626** to reach the walled garden **420** service having the specified plot number. Accordingly, the specified URL from the walled garden server will be served to the client **112**. In this case, the client **112** downloads and executes the JAVA, HTML, XML, and/or JAVASCRIPT code providing the service as described below. Preferably, the downloaded code is not persistently stored in the client **112**. If the WGPS **414** denies access, then it sends a HTTP status **407** response to the client **112** with an HTTP header indicating the reason for denying access. Typically, the client **112** will respond to this denial by requesting **616** a new ticket from the PS **438**.

FIG. **9** is a flow diagram illustrating transactions among a Walled Garden Server (WGS) **910**, the WGPS **414**, and the client **112** according to a preferred embodiment of the present invention when the WGS responses to an client request for a service. The WGS **910**, which may be any of the server types described above, sends **912** a message to the client **112** via HTTP. The message contains code in JAVASCRIPT invoking one or more of the functions in the APIs implemented in the shell **326** of the client **112**. Other embodiments of the present invention may use languages other than JAVASCRIPT or other invocation methods, such as MACROMEDIA FLASH, to call API functions.

The message from the WGS **910** to the client **112** necessarily passes through the WGPS **414**. Preferably, a proxy plug-in on the WGPS **414** traps all messages from WGS' to clients in order to attach an ACL to each message. When the WGPS **414** traps a message, it examines **914** the header provided by the WGS **910** for any potential security violations. For example, the WGPS **414** strips any improper headers off the message to protect against masquerading or spoofing by the WGS **910**. Then, the WGPS **414** looks up **916** the corresponding entry in the Walled Garden Permissions Table stored in the database **415** and retrieves the ACL for the given service, affiliation, and user agent. The WGPS **414** inserts **918** the ACL into the message from the WGS **910** to the client **112** as an HTTP header. In one embodiment of the present invention, the ACL is inserted into a "athmAPIAuth" header, although other headers or transport mechanisms can be used as well.

In addition, the WGPS **414** can place information in the header that further limits the permissions contained in the ACL. For example, the WGPS **414** can restrict the WGS **910** to accessing channel guide data for the current time only, for the next hour, for the next day or week, etc. Similarly, the WGPS **414** can restrict the WGS **910** to accessing channel guide data for only a certain channel or network. The WGPS **414** preferably implements these additional limitations by placing additional fields in the HTTP header. After the headers are inserted, the WGPS **414** passes **920** the message to the client **112**.

The shell **326** executing on the client **112** extracts the ACL, affiliations, and any other permissions from the headers and determines **922** whether the data grant the WGS **910** access to the API functions called by the attendant code. The shell **326** codifies the mapping from bit positions in the ACL to API functions and enforces the access control. If the ACL does not allow a called API function to be executed, then the shell **326** preferably returns **924** the FAIL_FUNCTION_NOT_AUTHORIZED message to the application or program that invoked the API function. Otherwise, the shell **326** returns **924** the result of the function invocation.

In summary, the present invention is an authentication and authorization method and system that lets individual users access one or more of the services within the walled garden **420**. The client **112** authentication procedure allows individual users to be authenticated. In addition, the GS **416**, PS **438**, and associated database **440** can authorize a unique set of access rights for each user. The WGPS **414** ensures that only authenticated and authorized users are allowed to access servers within the walled garden **420**. Moreover, the design of the system, including the ticket and shared secret key, provides an efficient implementation, thereby keeping a relatively light processing load on the GS **416** and PS **438**.

In addition, the present invention enhances the services provided by the walled garden **420** by allowing WGS' to access the APIs of the clients. The Walled Garden Permissions table stored in the database **415** of the WGPS **414** allows the access rights of a WGS to be controlled with a fine degree of granularity with respect to functions, time, and channels/networks.

By using the method and system described herein, a service provider or other entity can sell subscriptions or other forms of access rights to one or more services within the walled garden **420**. For example, an ISP can sell subscriptions to tiers of services, much like subscriptions to tiers of television channels are sold. In addition, the ISP can sell the right to access the client **112** APIs to the operators of the WGS'.

APPENDIX

Channel Navigation

Channel Up

Syntax

[Status]=TV.ChannelUp( )

Parameters

Status

The return status value from the ChannelUp method. The return status value will be one of the following:

SUCCESS—The tune completed successfully

FAIL_INVALID_CHANNEL—The tune failed because of an invalid channel number

FAIL_PARENTAL_CONTROL—The tune failed because of parental control and a valid Parental Control PIN was not entered

FAIL_CHANNEL_NOT_AUTHORIZED—The tune failed because the channel was not authorized by the CA system

FAIL_FUNCTION_NOT_AUTHORIZED—The tune failed because the function invocation was not authorized by the CA system

FAIL_NOT_PURCHASED—The tune failed because the channel carried an IPPV event and it was not purchased

Channel Down

Syntax

[Status]=TV.ChannelDown( )

Parameters

Status

The return status value will be one of the following:

SUCCESS—The tune completed successfully

FAIL_INVALID_CHANNEL—The tune failed because of an invalid channel number

FAIL_PARENTAL_CONTROL—The tune failed because of parental control and a valid Parental Control PIN was not entered

FAIL_CHANNEL_NOT_AUTHORIZED—The tune failed because the channel was not authorized by the CA system

FAIL_FUNCTION_NOT_AUTHORIZED—The tune failed because the function invocation was not authorized by the CA system

FAIL_NOT_PURCHASED—The tune failed because the channel carried an IPPV event and it was not purchased

Direct Channel Specification

Syntax

[Status]=TV.TuneChannel(Channel)

Parameters

Channel

A numeric value that specifies the channel number.

Status

The return status value will be one of the following:

SUCCESS—The tune completed successfully

FAIL_INVALID_CHANNEL—The tune failed because of an invalid channel number

FAIL_PARENTAL_CONTROL—The tune failed because of parental control and a valid Parental Control PIN was not entered

FAIL_CHANNEL_NOT_AUTHORIZED—The tune failed because the channel was not authorized by the CA system

FAIL_FUNCTION_NOT_AUTHORIZED—The tune failed because the function invocation was not authorized by the CA system

FAIL_NOT_PURCHASED—The tune failed because the channel carried an IPPV event and it was not purchased

Remark

The TuneChannel method tunes to the specified channel number.

Get Current Channel

Syntax

[Status]=TV.GetCurrentChannel(Channel)

Parameters

Channel

A numeric value that returns the current channel number.

Status

The return status value from the TuneChannel method. The return status value will be one of the following:

SUCCESS—The function completed successfully

FAIL_INVALID_CHANNEL—The tune failed because of an invalid channel number

Channel Map Access

Channel map access allows applications to inquiry about the details of the channel line-up (aka channel map) currently available in the set-top. Access to the service information, i.e. station call letters and network identifiers are available through the channel map access.

Syntax

[Status]=ChannelMap.ChannelToNetwork (Channel, Network, Station)

Parameters

Channel

A numeric value that specifies the channel number.

Network

A string value that returns the broadcast network name.

Station

A string value that returns the local station name.

Status

The return status value will be one of the following:

SUCCESS—The function completed successfully

FAIL_INVALID_CHANNEL—The function failed because of an invalid channel number

FAIL_FUNCTION_NOT_AUTHORIZED—The function failed because the function invocation was not authorized by the CA system

Syntax

[Status]=ChannelMap.NetworkToChannel (Network, Channel, Station)

Parameters

Network

A string value that specifies the broadcast network name.

Channel

A numeric value that returns the channel number.

Station

A string value that returns the local station name.

Status

The return status value from the NetworkToChannel method. The return status value will be one of the following:

SUCCESS—The function completed successfully

FAIL_INVALID_CHANNEL—The function failed because of an invalid channel number

FAIL_FUNCTION_NOT_AUTHORIZED—The function failed because the function invocation was not authorized by the CA system

Remark

The NetworkToChannel method returns the channel number and station name for the specified network name. If there are more than one channel that carries the specified network name, only the first channel map entry is returned.

Syntax

[Status]=ChannelMap.ChannelRange (MinChannel, MaxChannel)

Parameters

MinChannel

A numeric value that returns the minimum channel number.

MaxChannel

A numeric value that returns the maximum channel number.

Status

The return status value from the ChannelRange method. The return status value will be one of the following:

SUCCESS—The function completed successfully

FAIL_FUNCTION_NOT_AUTHORIZED—The function failed because the function invocation was not authorized by the CA system

Remark

The ChannelRange method returns the minimum and maximum channel numbers supported by the channel map provided by the cable system.

EPG Data Access

EPG Data Access allows applications to inquiry details of the program guide data. The EPG Data Access is controlled by the following attributes:

- Access to schedule information (program name, start time, end time, and rating)
- Access to editorial information (program description, etc.)
- Access to PPV information (pricing, etc.)
- Dividing up schedule information (current time, next hour, next day, etc.)
- Control access on a source by source basis (only access guide data for a particular network)

Schedule Information

Syntax

[Status]=EPG.GetScheduleInfo (Channel, RelativeTime, ProgramName, StartTime, EndTime, Rating)

Parameters

Channel

A numeric value that specifies the channel number for which schedule information is requested

RelativeTime

A numeric value that specifies the relative time in minutes from the current time for which schedule information is requested

ProgramName

A string value that returns the program name of the program that occurs on the specified channel at the specified relative time.

StartTime

A numeric value that returns the start time of the program that occurs on the specified channel at the specified relative time.

EndTime

A numeric value that returns the end time of the program that occurs on the specified channel at the specified relative time.

Status

The return status value will be one of the following:

SUCCESS—The function completed successfully

FAIL_FUNCTION_NOT_AUTHORIZED—The function failed because the function invocation was not authorized by the CA system

FAIL_TIME_NOT_AUTHORIZED—The function failed because access to schedule information at the specified relative time was not authorized by the CA system

FAIL_CHANNEL_NOT_AUTHORIZED—The function failed because access to schedule information on the specified channel was not authorized by the CA system

Remark

The GetScheduleInfo method returns the program name, start time, end time, and rating of the program that is on the specified channel at the specified relative time.

Program Information

Syntax

[Status]=EPG.GetProgramInfo (Channel, RelativeTime, ProgramDescription)

Parameters

Channel

A numeric value that specifies the channel number for which schedule information is requested

Relative Time

A numeric value that specifies the relative time in minutes from the current time for which schedule information is requested

ProgramDescription

A string value that returns the detailed program description of the program that occurs on the specified channel at the specified relative time.

Status

The return status value from the GetProgramInfo method. The return status value will be one of the following:

SUCCESS—The function completed successfully

FAIL_FUNCTION_NOT_AUTHORIZED—The function failed because the function invocation was not authorized by the CA system

FAIL_TIME_NOT_AUTHORIZED—The function failed because access to schedule information at the specified relative time was not authorized by the CA system

FAIL_CHANNEL_NOT_AUTHORIZED—The function failed because access to schedule information on the specified channel was not authorized by the CA system

Remark

The GetProgramInfo method returns the detailed program description of the program that is on the specified channel at the specified relative time.

Pay-Per-View Information

Syntax

[Status]=EPG.GetPPVInfo (Channel, RelativeTime, ISPPV, Price)

Parameters

Channel

A numeric value that specifies the channel number for which schedule information is requested

RelativeTime

A numeric value that specifies the relative time in minutes from the current time for which schedule information is requested

IsPPV

A boolean value that indicates if specified program that occurs on the specified channel at the specified relative time is a PPV event. A value of TRUE indicates that it is a PPV event, a value of FALSE indicates that it is not a PPV event.

Price

A string value that returns the price of the PPV event that occurs on the specified channel at the specified relative time. A value of NULL is returned if the specified program is not a PPV event.

Status

The return status value from the GetScheduleInfo method. The return status value will be one of the following:

SUCCESS—The function completed successfully

FAIL_FUNCTION_NOT_AUTHORIZED—The function failed because the function invocation was not authorized by the CA system

FAIL_TIME_NOT_AUTHORIZED—The function failed because access to schedule information at the specified relative time was not authorized by the CA system

FAIL_CHANNEL_NOT_AUTHORIZED—The function failed because access to schedule information on the specified channel was not authorized by the CA system

Remark

The GetPPVInfo method returns the PPV status and price of the program that is on the specified channel at the specified relative time.

Instantiation Of Standard UI Elements

The Channel Banner

Syntax

[Status]=UI.DisplayChannelBanner ( )

Parameters

Status

The return status value will be one of the following:

SUCCESS—The function completed successfully

FAIL_FUNCTION_NOT_AUTHORIZED—The function failed because the function invocation was not authorized by the CA system

Remark

The DisplayChannelBanner method causes the NavShell Foundation to display the NavShell channel banner as an overlay. The channel banner will automatically be taken down after its time-out period has been reached.

The Extras Menu

Syntax

[Status]=UI.DisplayExtrasMenu (NumberOfEntries, MenuEntries, EntrySelected)

Parameters

NumberOfEntries

A numeric value that specifies the number of menu entries contained in the MenuEntries array

MenuEntries

An array of string values that specify the text for each of menu entries to be displayed in the Extras Menu

EntrySelected

A numeric value that returns the index of menu entry selected by the viewer from the MenuEntries array

Status

The return status value from the DisplayExtrasMenu method. The return status value will be one of the following:

SUCCESS—The function completed successfully

FAIL_FUNCTION_NOT_AUTHORIZED—The function failed because the function invocation was not authorized by the CA system

FAIL_VIEWER_EXIT—The function failed because the viewer pressed the EXIT key

FAL_DIALOG_TIMEOUT—The function failed because the time-out period for the viewer identification dialog box expires before the viewer makes a selection

Remark

The DisplayExtrasMenu method causes the NavShell Foundation to display the NavShell extras menu as an overlay. The viewer may then select among the entries specified by the MenuEntries array. When the viewer selects one of the menu entries, the corresponding index is returned. The extras menu will automatically be taken down after its time-out period has been reached.

Viewer Accounts

Number of Viewer Accounts

Syntax

[Status]=GetNumberOfViewers (NumberOfViewers)

Parameters

NumberOfViewers

A numeric value that returns the number of viewer accounts that have been defined for the household.

Status

The return status value will be one of the following:

SUCCESS—The function completed successfully

FAIL_FUNCTION_NOT_AUTHORIZED—The function failed because the function invocation was not authorized by the CA system

Remark

The GetNumberOfViewers method returns the number of viewer accounts defined for the set-top household. A return value of one indicates that only the Default_Viewer is defined for the household.

Viewer Names

Syntax

[Status]=GetViewerName (ViewerNumber, ViewerName)

Parameters

ViewerNumber

A numeric value that specifies which viewer name is being requested. This value must be in the range 1 to the value returned by GetNumberOfViewers inclusive.

ViewerName

A string value that returns the name for the specified viewer.

Status

The return status value from the GetViewerName method. The return status value will be one of the following:

SUCCESS—The function completed successfully

FAIL_FUNCTION_NOT_AUTHORIZED—The function failed because the function invocation was not authorized by the CA system

FAIL_INDEX_NOT_VALID—The function failed because specified ViewerNumber was outside the valid range

Remark

The GetViewerName method returns the viewer name for the specified viewer.

Viewer Privileges

Syntax

[Status]=GetViewerPrivileges (ViewerNumber, TVAccess, WebAccess, IPPVEnabled, EmailEnabled, WalletEnabled)

Parameters

ViewerNumber

A numeric value that specifies which viewer name is being requested. This value must be in the range 1 to the value returned by GetNumberOfViewers inclusive.

TVAccess

A boolean value that returns whether the specified viewer's TV viewing access is restricted

WebAccess

A boolean value that returns whether the specified viewer's Web browsing access is restricted

Status

The return status value from the GetViewerPrivileges method. The return status value will be one of the following:

SUCCESS—The function completed successfully

FAIL_FUNCTION_NOT_AUTHORIZED—The function failed because the function invocation was not authorized by the CA system

FAIL_INDEX_NOT_VALID—The function failed because specified ViewerNumber was outside the valid range

Remark

The GetViewerPrivileges method returns the viewer privileges for the specified viewer.

Viewer Parental Controls

Syntax

[Status]=GetViewerParentalControls (ViewerNumber, TVRating, MovieRating, WebAccess, EmallAccess)

Parameters

ViewerNumber

A numeric value that specifies which viewer name is being requested. This value must be in the range 1 to the value returned by GetNumberOfViewers inclusive.

TVRating

A string value that returns the specified viewer's TV rating parental control setting

MovieRating

A string value that returns the specified viewer's MPAA rating parental control setting

Status

The return status value from the GetViewerParentalControls method. The return status value will be one of the following:

SUCCESS—The function completed successfully

FAIL_FUNCTION_NOT_AUTHORIZED—The function failed because the function invocation was not authorized by the CA system

FAIL_INDEX_NOT_VALID—The function failed because specified ViewerNumber was outside the valid range

Remark

The GetViewerParentalControls method returns the viewer parental control settings for the specified viewer.

Viewer Identification

Syntax

[Status]=ViewerIdentification (Message, Viewer)

Parameters

Message

A string value that specifies the message text displayed in the viewer identification dialog box.

Viewer

A string value that returns the viewer name entered by the viewer. The viewer name will be one of the viewer accounts associated with the household.

Status

The return status value from the ViewerIdentification method. The return status value will be one of the following:

SUCCESS—The function completed successfully

FAIL_FUNCTION_NOT_AUTHORIZED—The function failed because the function invocation was not authorized by the CA system

FAIL_INVALID_PIN—The function failed because the viewer entered an invalid PIN for the selected viewer

FAIL_VIEWER_EXIT—The function failed because the viewer pressed the EXIT key

FAIL_DIALOG_TIMEOUT—The function failed because the time-out period for the viewer identification dialog box expires before the viewer makes a selection

Remark

The ViewerIdentification method displays a viewer identification dialog box with the specified text message. The viewer then can select the desired viewer from the list of available viewers associated with the household and enter the appropriate PIN for that viewer to identify himself. The ViewerIdentification method then returns the name of the viewer selected. The viewer can also press the EXIT key to terminate the viewer identification dialog box without selecting a viewer, or they can let the dialog box time-out, or enter an invalid PIN. These conditions are returned in the Status return value. In households where only a single viewer account has been created, the viewer does not have the option of selecting among multiple viewers, but must still enter a valid PIN. The viewer name returned when the correct PIN is entered is Default_Viewer.

E-wallet and Buy-sequence

E-Wallet Purchase

Syntax

[Status]=EwalletPurchase (NumberOfPurchaseOptions, PurchaseOptions, SelectedOption, Buyer, Shipto)

Parameters

NumberOfPurchaseOptions

A numeric value that specifies the number of purchase options contained in the PurchaseOptions array.

PurchaseOptions

An array of string values that specifies the message text for each purchase option that is available in the E-wallet purchase transaction dialog box. The first entry in this array is the default option displayed.

SelectedOption

A numeric value that returns the index of purchase option selected by the viewer from the PurchaseOptions array.

Status

The return status value will be one of the following:

SUCCESS—The function completed successfully

FAIL_FUNCTION_NOT_AUTHORIZED—The function failed because the function invocation was not authorized by the CA system

FAIL_INVALID_PIN—The function failed because the viewer entered an invalid PIN for the selected viewer

FAIL_VIEWER_EXIT—The function failed because the viewer pressed the EXIT key

FAIL_DIALOG_TIMEOUT—The function failed because the time-out period for the viewer identification dialog box expires before the viewer makes a selection

Remark

The EwalletPurchase method displays an E-Wallet purchase transaction dialog box with the specified array of purchase options. The viewer then can select the desired purchase option from the list of available purchase options. The EwalletPurchase method then returns the index of the viewer selected purchase option, the name of the viewer that is performing the purchase, and the name of the person to which the purchase will be shipped. The viewer can also press the EXIT key to terminate the E-Wallet purchase transaction dialog box without selecting a purchase option, or they can let the dialog box time-out, or enter an invalid PIN. These conditions are returned in the Status return value. In households where only a single viewer account has been created, the viewer does not have the option of selecting

among multiple viewers who is performing the purchase, but must still enter a valid PIN. The viewer name returned when the correct PIN is entered is Default_Viewer.

Reminders

Set Reminder

Syntax

[Status]=Rem.SetReminder (RelativeDay, TimeOfDay, ReminderMessage, ForUser, ReminderURL)

Parameters

RelativeDay

A numeric value that specifies the number of days in the future at which the reminder is to be triggered.

TimeOfDay

A numeric value that specifies the time of day on the day specified by RelativeDay at which the reminder is to be triggered.

ReminderMessage

A string value that specifies the message text displayed in the TV Notice dialog box.

ReminderURL

A string value that specifies the URL that is accessed when the viewer selects "Yes" in the TV Notice dialog box.

ForUser

A string value that specifies the viewer name for whom the reminder is intended. The viewer name must be one of the viewer accounts associated with the household.

Status

The return status value from the SetReminder method. The return status value will be one of the following:

SUCCESS—The function completed successfully

FAIL_FUNCTION_NOT_AUTHORIZED—The function failed because the function invocation was not authorized by the CA system

FAIL_INVALID_VIEWER—The function failed because the specified viewer name is not a valid viewer of the household

FAIL_REMINDER_CONFLICT—The function failed because there is an existing reminder set that conflicts with the specified time for the requested reminder.

Remark

The SetReminder method sets a reminder for the specified viewer that displays the specified reminder message when the reminder is triggered. The specified URL is displayed if the viewer elects to act on the reminder when it occurs. The reminder is displayed as a TV Notice.

Display TV Notice

This method allows the content provider to simulate a reminder by directly displaying the TV Notice dialog box immediately, rather than setting a reminder and waiting for the reminder to be triggered.

Syntax

[Status]=Rem.DisplayTVNotice (NoticeMessage, NoticeURL)

Parameters

NoticeMessage

A string value that specifies the message text displayed in the TV Notice dialog box.

NoticeURL

A string value that specifies the URL that is accessed when the viewer selects "Yes" in the TV Notice dialog box.

Status

The return status value will be one of the following:

SUCCESS—The function completed successfully

FAIL_FUNCTION_NOT_AUTHORIZED—The function failed because the function invocation was not authorized by the CA system

FAIL_INVALID_VIEWER—The function failed because the specified viewer name is not a valid viewer of the household

Remark

The SetReminder method sets a reminder for the specified viewer that displays the specified reminder message when the reminder is triggered. The specified URL is displayed if the viewer elects to act on the reminder when it occurs. The reminder is displayed as a TV Notice.

Printing

Syntax

[Status]=PrintPage ( )

Parameters

Status

The return status value from the DisplayTVNotice method. The return status value will be one of the following:

SUCCESS—The function completed successfully

FAIL_FUNCTION_NOT_AUTHORIZED—The function failed because the function invocation was not authorized by the CA system

FAIL_PRINTER_ERROR—The function failed because of an error on the printer

Remark

The PrintPage method prints the current page displayed on the TV.

We claim:

1. A system for restricting access to a walled garden having network-based services identified by plot numbers and provided on a private network comprising:

a plurality of servers coupled to the walled garden for providing network-based services;

a gateway server coupled to a network for issuing a ticket to a client, the ticket specifying with a set of bits the network-based services to which a user of the client has access; and

a walled garden proxy server coupled to the walled garden and the networks for receiving from the client the ticket and a request to access a network-based services identified by a plot number, and for determining from the ticket whether to grant the client access to the network-based services by using the plot number as an index into the set of bits in the ticket.

2. The system of claim 1, further comprising:

a first database for holding information describing access rights for a plurality of users; and

a policy server in communication with the gateway server and the first database for accessing the first database responsive to requests received from the gateway server and retrieving information specifying the network-based services to which the user has access.

3. The system of claim 1, further comprising:

a program module executable by the gateway server for authenticating the user;

wherein the gateway server issues the ticket in response to a positive authentication of the user.

4. The system of claim 1, wherein the walled garden and the networks are logically derived from a single physical network.

5. The system of claim 1 wherein the gateway server issues the client a second ticket specifying Internet-based servers to which the user has access and further comprising:

an Internet proxy server coupled to the network for receiving from the client a request to access Interet-based services and the second ticket issued by the gateway server and determining from the second ticket whether to grant the client access to the Internet-based services.

6. The system of claim 1, further comprising:

a program module executable by the gateway server for encrypting the ticket issued to the client; and

a program module executable by the walled garden proxy server for decrypting the ticket received from the client.

7. The system of claim 6, further comprising:

a keymaster in secure communication with the gateway server and the walled garden proxy server for issuing keys for encrypting and decrypting the ticket.

8. The system of claim 1, wherein the plurality of servers providing network-based services comprise at least one server selected from the group consisting of:

a first application server directly coupled to the walled garden;

a second application server directly coupled to the walled garden and coupled via a dedicated connection to a remote application database for supporting the network-based services provided by the second application server;

a first remote application server coupled to the walled garden via a virtual network; and

a front end server coupled directly to the walled garden for providing a link to a second remote application server.

9. A method of restricting access to a walled garden having network-based services identified by plot numbers and available on a private network, comprising the steps of:

receiving a request from a client to access a plot number of a network-based service available on the walled garden;

receiving a ticket from the client describing with a set of bits the network-based services to which a user of the client has access;

determining from the ticket whether the user has access to the requested network-based service by using the plot number as an index into the set of bits in the ticket; and

responsive to a positive determination that the user has access to the requested network-based service, allowing the client to access the network-based service.

10. The method of claim 9, further comprising the step of:

responsive to a negative determination that the user has access to the requested network-based service, denying the client access to tile network service.

11. The method of claim 9, wherein the request from the client to access a network-based service available on the walled garden is not accompanied by a ticket and further comprising the step of:

denying the client access to the network service.

12. The method of claim 9, further comprising the steps of:

receiving a request from the client to issue the ticket;

receiving authentication information from the user of the client;

authenticating the user of the client with the authentication information;

responsive to a successful authentication of the user, generating the ticket; and

transmitting the generated ticket to the client.

13. The method of claim 12, wherein the step of generating the ticket comprises the steps of:

storing information in the ticket indicating a box ID of the client;

storing information in the ticket indicating an expiration date for the ticket; and

storing information in the ticket indicating the network services with which the ticket is affiliated.

14. The method of claim 12, wherein the step of generating the ticket further comprises the step of:

encrypting the ticket.

15. The method of claim 11, wherein the ticket received from the client is encrypted and the determining step comprises the step of:

decrypting the ticket.

16. A system for restricting access by clients to a walled garden providing a plurality of services, the services identified by plot numbers, comprising:

a gateway server for authenticating users of the clients requesting access to the plurality of services by plot numbers and issuing tickets to the clients responsive to positive authentications of the users, the tickets including sets of bits granting the clients access to at least one of the plurality of services provided by the walled garden; and

a walled garden proxy server for receiving the requests from the clients to access the plurality of services provided by the walled garden, wherein the walled garden proxy server grants a client request to access a service if the request includes a ticket granting access to the requested service as determined by using the plot number of the service as an index into the set of bits in the ticket.

17. The system of claim 16, further comprising:

a database in communication with the gateway server for identifying access rights to the plurality of services in the walled garden of the users of the clients.

18. The system of claim 16, further composing:

a keymaster in secure communication with the gateway server and the walled garden proxy server for issuing secret keys;

wherein the gateway server encrypts at least portions of issued tickets using the secret keys and the walled garden proxy server decrypts the encrypted portions of the tickets using the secret keys.

19. The system of claim 16, wherein the walled garden comprises at least one server selected from the group consisting of:

a first application server directly coupled to the walled garden;

a second application server directly coupled to the walled garden and coupled via a dedicated connection to a remote application database for supporting the network services provided by the second application server;

a first remote application server coupled to the walled garden via a virtual network; and

a front end server coupled directly to the walled garden for providing a link to a second remote application server.

20. The system of claim 16, wherein the gateway server authenticates users of the clients requesting access to sites

on the Internet and issues tickets to the clients responsive to positive authentications of the users, the tickets granting the clients access to the sites on the Internet, the system further comprising:

an Internet server for receiving requests from the clients to access sites on the Internet, wherein the Internet server grants a client request to access a site on the Internet if the request includes a ticket granting access to the requested site.

21. The system of claim **16**, wherein the walled garden proxy server restricts access to a plurality of walled gardens and wherein the tickets issued by the gateway server specify the walled garden of the plurality of walled gardens with which the ticket is affiliated.

22. The system of claim **1**, further comprising:

a second database in communication with the proxy server for identifying invalid tickets.

23. The method of claim **9**, wherein the determining step comprises the step of:

checking a database of invalid tickets to determine whether the received ticket is invalid.

24. The system of claim **22**, wherein the database holds at least one decryption key for decrypting an encrypted ticket received from the client.

25. The method of claim **23**, further comprising the step of:

checking the database to determine whether the received ticket is affiliated with the network-based services available on the walled garden; wherein the client is denied access to the network services responsive to a negative determination that the received ticket is affiliated with the network-based services available on the walled garden.

* * * * *

US006263432B1

(12) **United States Patent**
Sasmazel et al.

(10) **Patent No.: US 6,263,432 B1**
(45) **Date of Patent: *Jul. 17, 2001**

(54) **ELECTRONIC TICKETING, AUTHENTICATION AND/OR AUTHORIZATION SECURITY SYSTEM FOR INTERNET APPLICATIONS**

(75) Inventors: **Levent M D Sasmazel**, Holmdel; **David H. Schneider**, Manalapan, both of NJ (US)

(73) Assignee: **NCR Corporation**, Dayton, OH (US)

(*) Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **08/944,754**

(22) Filed: **Oct. 6, 1997**

(51) **Int. Cl.**[7] **G06F 1/24**; G06F 9/00
(52) **U.S. Cl.** **713/100**
(58) **Field of Search** 713/100

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,481,720	1/1996	Loucks et al.	713/201
5,535,276	7/1996	Ganesan	713/155
5,544,322	8/1996	Cheng et al.	709/229
5,560,008	9/1996	Johnson et al.	713/201
5,706,427	1/1998	Tabuki	713/201

FOREIGN PATENT DOCUMENTS

0 695 985 2/1996 (EP) .

OTHER PUBLICATIONS

(1) B. Schneier, "Applied Cryptography, 2nd Ed.: Protocols, Algorithms, and Source Code in C", 1996.
Bruce Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, 1994.*

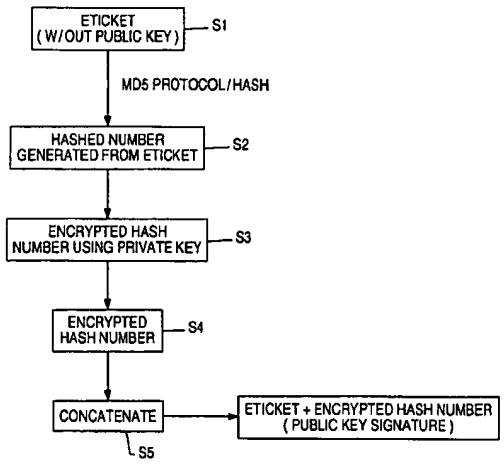
* cited by examiner

Primary Examiner—George B. Davis
(74) *Attorney, Agent, or Firm*—Lowe, Hauptman, Gopstein Gilman & Berner, LLP

(57) **ABSTRACT**

A computer program memory stores computer instructions for securing data transmitted over a system, such as the Internet, enabling a user to be authenticated and authorized for a requested operation. An "eticket" architecture (including identification information) is generated by an authentication server. The information in the eticket is hashed using, for example, a Message Digest Protocol, and a hash number is generated. The hash number is then encrypted using a private key, and the identification information in the eticket and the encrypted hash number are concatenated to generate a completed "eticket" architecture. The "eticket" may then be transmitted over the Internet (i.e., a non-secure environment) from server to server without having the information in the "eticket" altered, and without having to "reauthenticate" the user at each server.

22 Claims, 17 Drawing Sheets



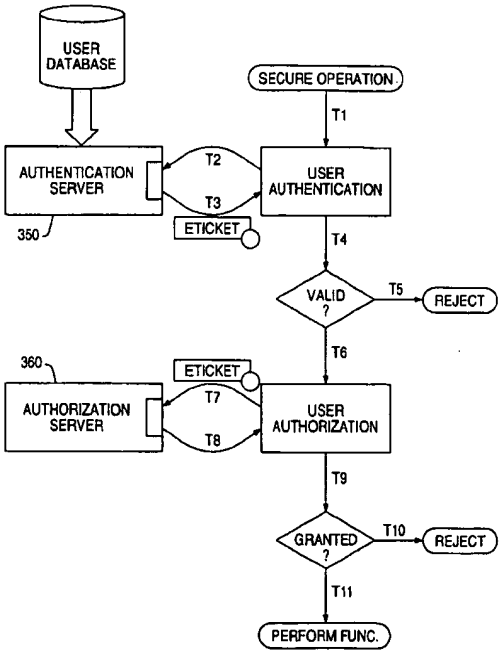
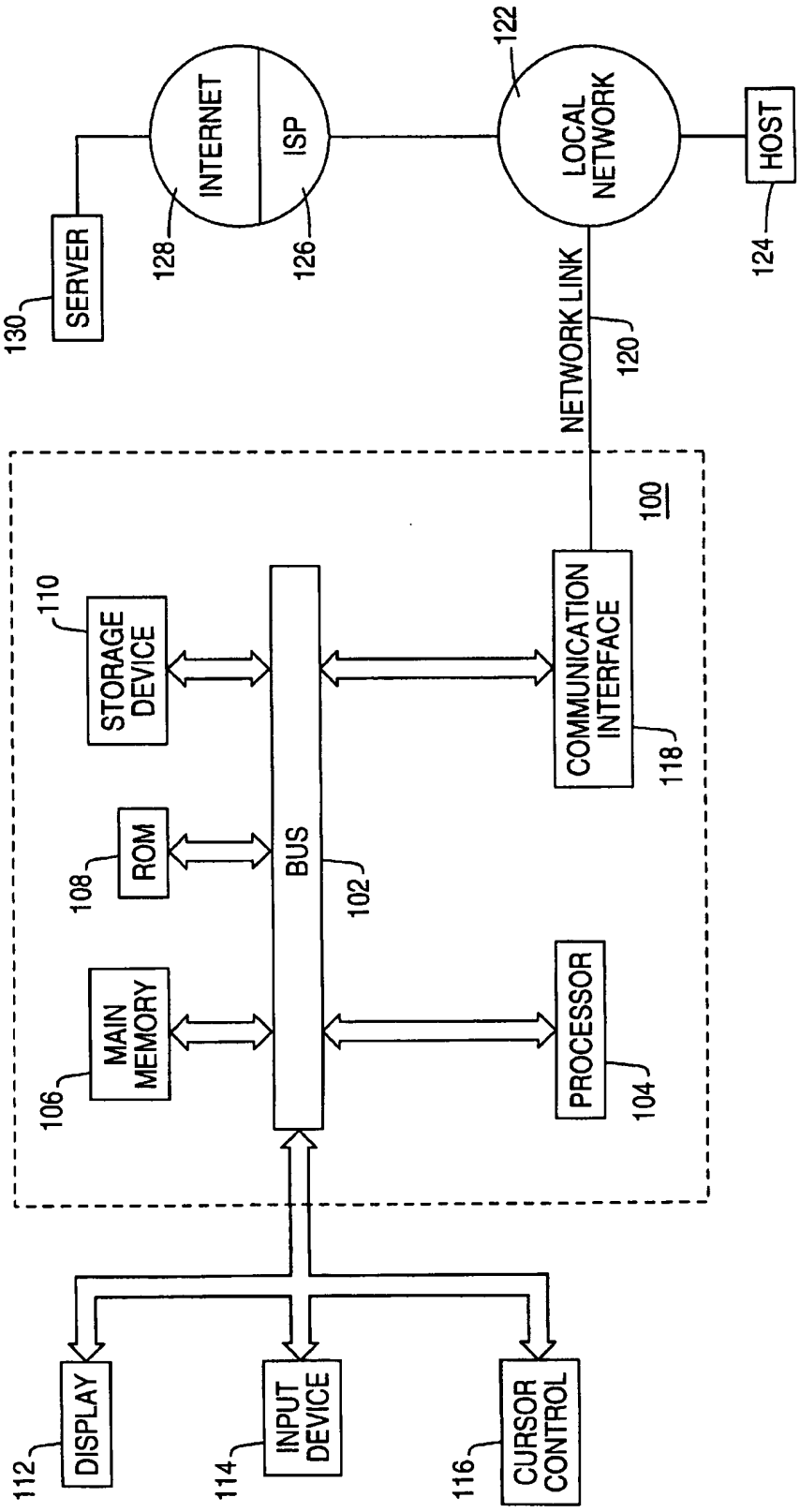


FIG. 1


INTERNET
ISP
LOCAL NETWORK
HOST
SERVER
128
126
122
124
130
NETWORK LINK
120
100
STORAGE DEVICE
110
COMMUNICATION INTERFACE
118
ROM
108
BUS
102
MAIN MEMORY
106
PROCESSOR
104
DISPLAY
112
INPUT DEVICE
114
CURSOR CONTROL
116

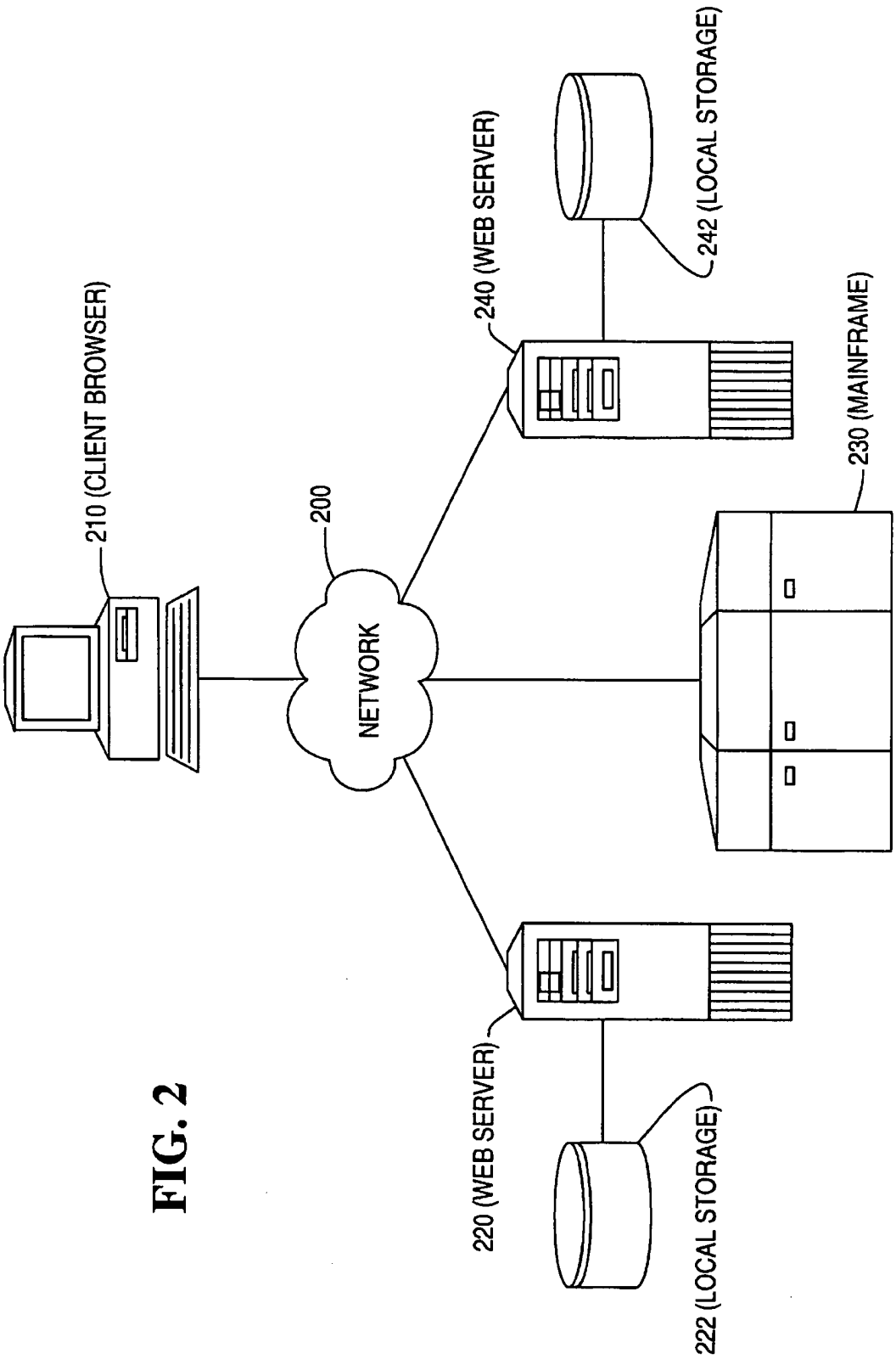

210 (CLIENT BROWSER)
200
NETWORK
240 (WEB SERVER)
242 (LOCAL STORAGE)
230 (MAINFRAME)
220 (WEB SERVER)
222 (LOCAL STORAGE)

FIG. 2

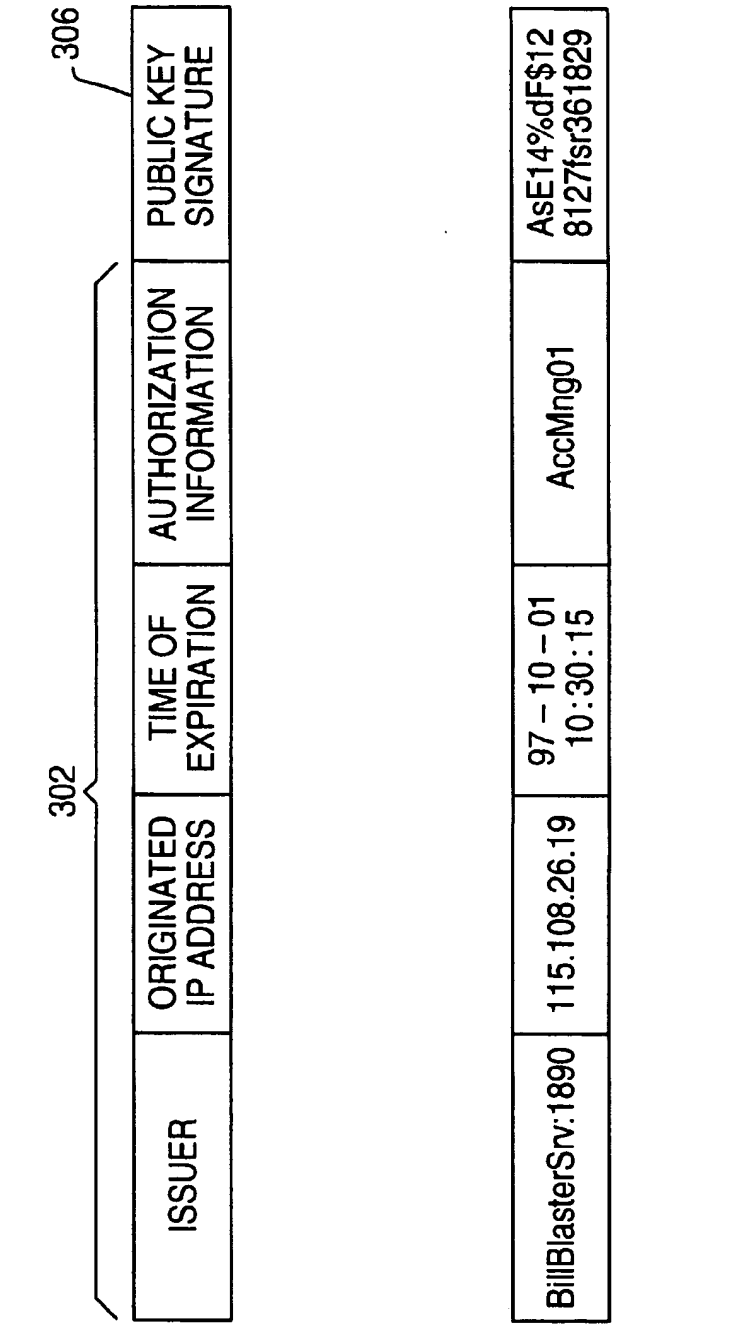

306
302
ISSUER
ORIGINATED IP ADDRESS
TIME OF EXPIRATION
AUTHORIZATION INFORMATION
PUBLIC KEY SIGNATURE
BillBlasterSrv:1890
115.108.26.19
97 – 10 – 01 10:30:15
AccMng01
AsE14%dF$12 8127fsr361829

FIG. 3

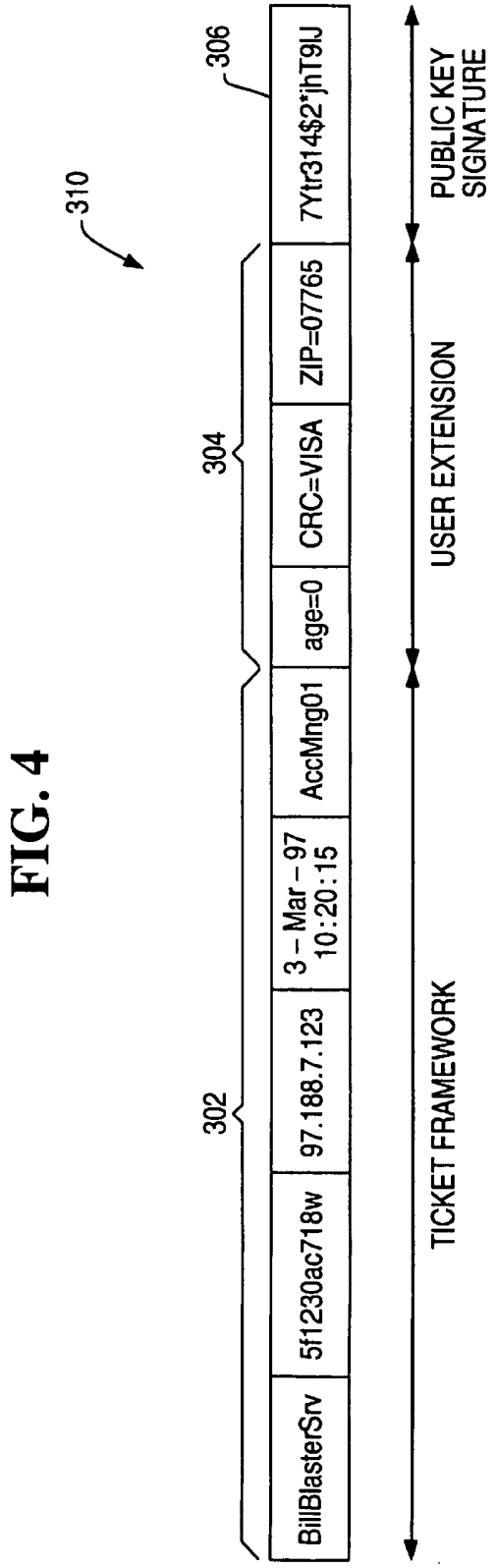

310
306
304
302
BillBlasterSrv
5f1230ac718w
97.188.7.123
3 – Mar – 97
10:20:15
AccMng01
age=0
CRC=VISA
ZIP=07765
7Ytr314$2*jhT9J
TICKET FRAMEWORK
USER EXTENSION
PUBLIC KEY
SIGNATURE

FIG. 4

FIG. 5

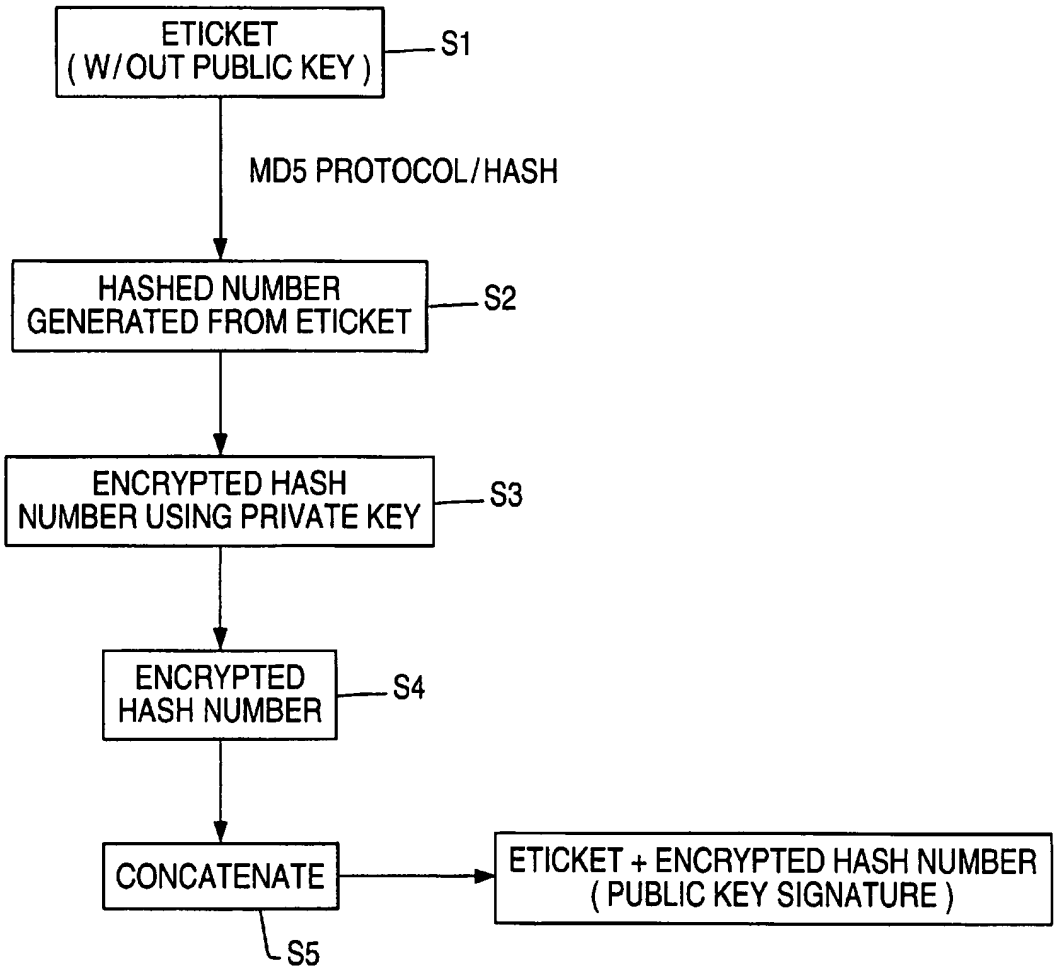

ETICKET
(W/OUT PUBLIC KEY)
S1
MD5 PROTOCOL/HASH
HASHED NUMBER
GENERATED FROM ETICKET
S2
ENCRYPTED HASH
NUMBER USING PRIVATE KEY
S3
ENCRYPTED
HASH NUMBER
S4
CONCATENATE
S5
ETICKET + ENCRYPTED HASH NUMBER
(PUBLIC KEY SIGNATURE)

FIG. 6

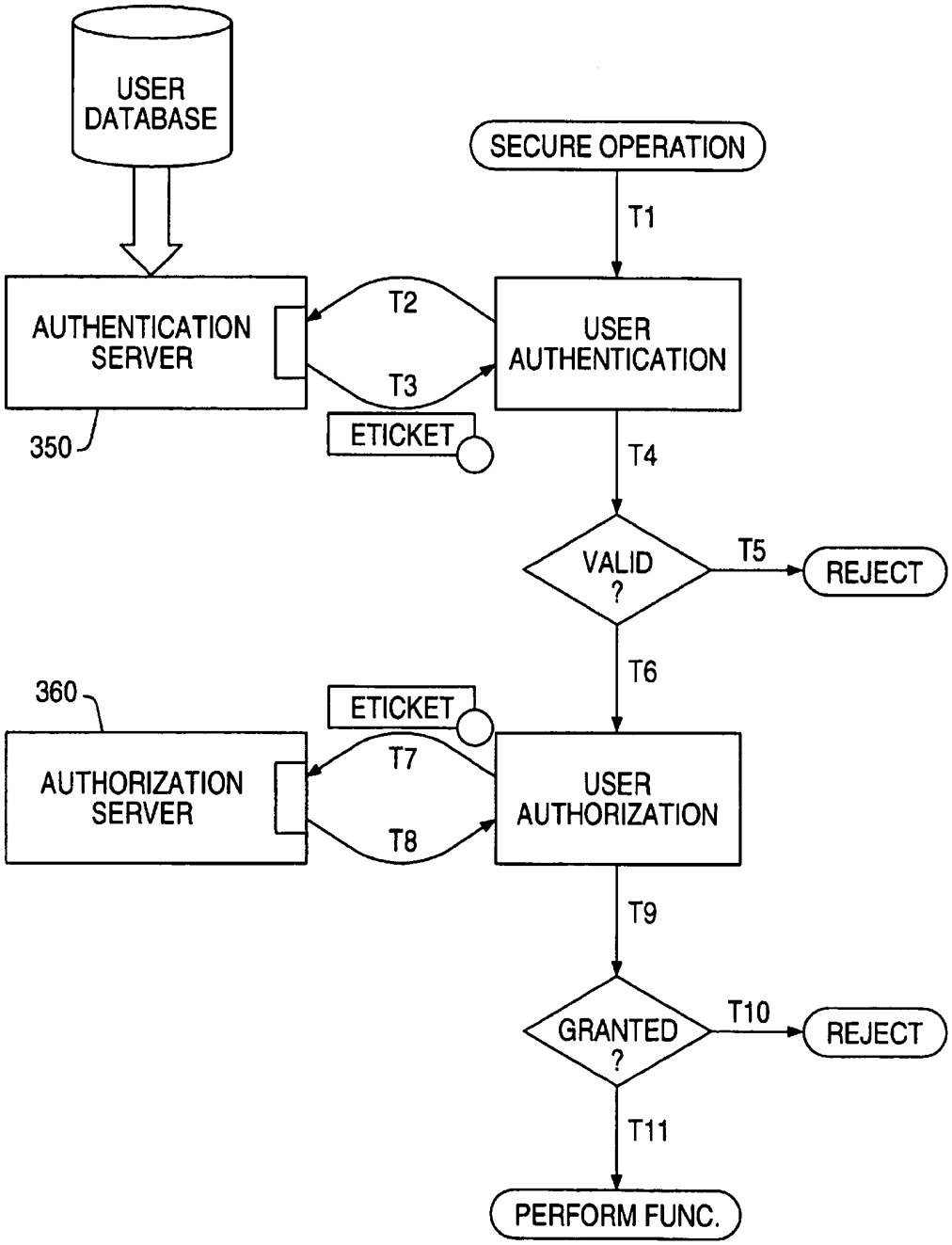
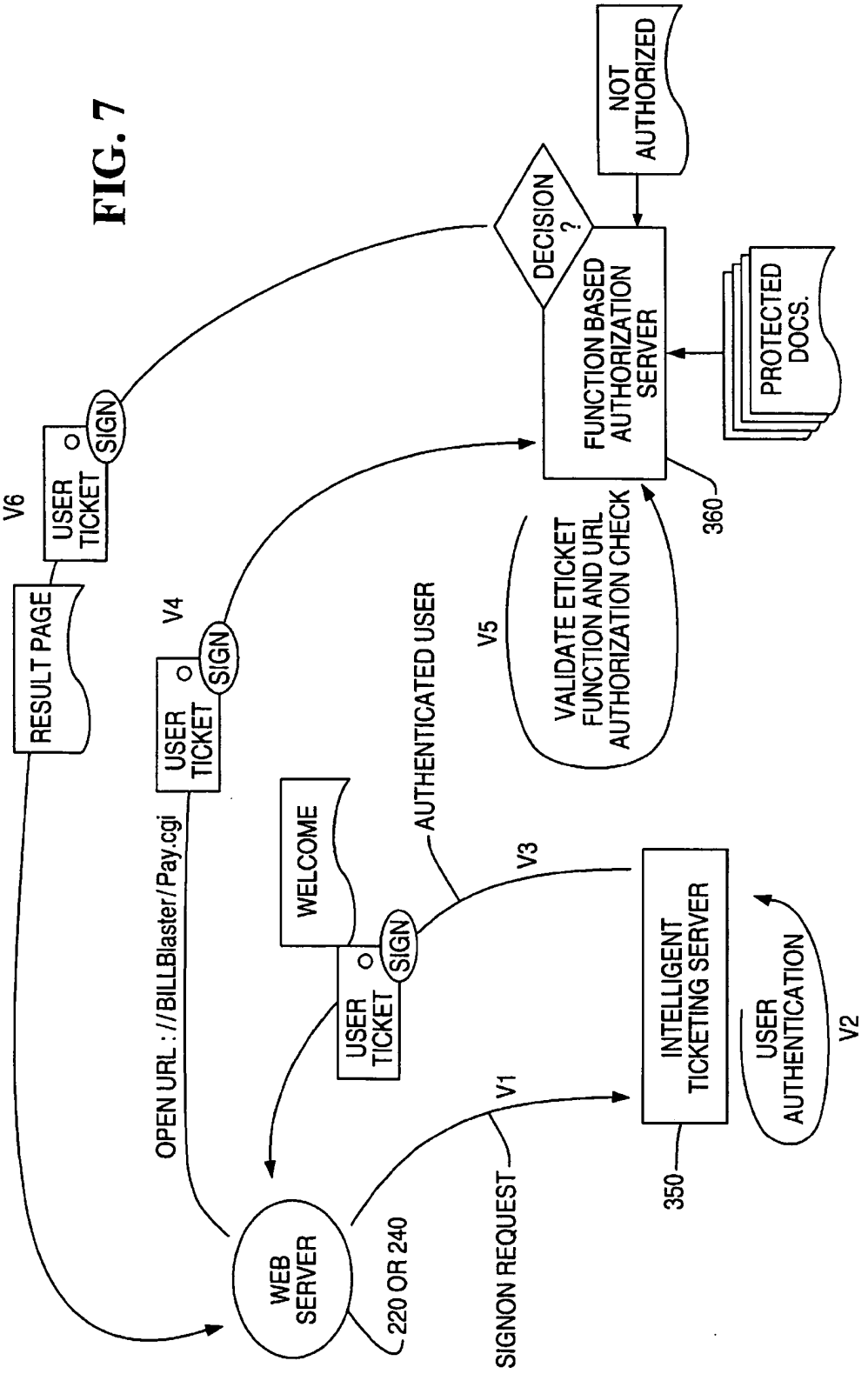

USER DATABASE
SECURE OPERATION
T1
AUTHENTICATION SERVER
T2
T3
USER AUTHENTICATION
350
ETICKET
T4
VALID ?
T5
REJECT
T6
360
ETICKET
AUTHORIZATION SERVER
T7
T8
USER AUTHORIZATION
T9
GRANTED ?
T10
REJECT
T11
PERFORM FUNC.

FIG. 7


NOT AUTHORIZED
DECISION ?
FUNCTION BASED AUTHORIZATION SERVER
PROTECTED DOCS.
360
V6
USER TICKET
SIGN
RESULT PAGE
V4
USER TICKET
SIGN
V5
VALIDATE ETICKET FUNCTION AND URL AUTHORIZATION CHECK
AUTHENTICATED USER
WELCOME
USER TICKET
SIGN
V3
INTELLIGENT TICKETING SERVER
USER AUTHENTICATION
V2
OPEN URL://BILLBlaster/Pay.cgi
V1
WEB SERVER
220 OR 240
SIGNON REQUEST
350

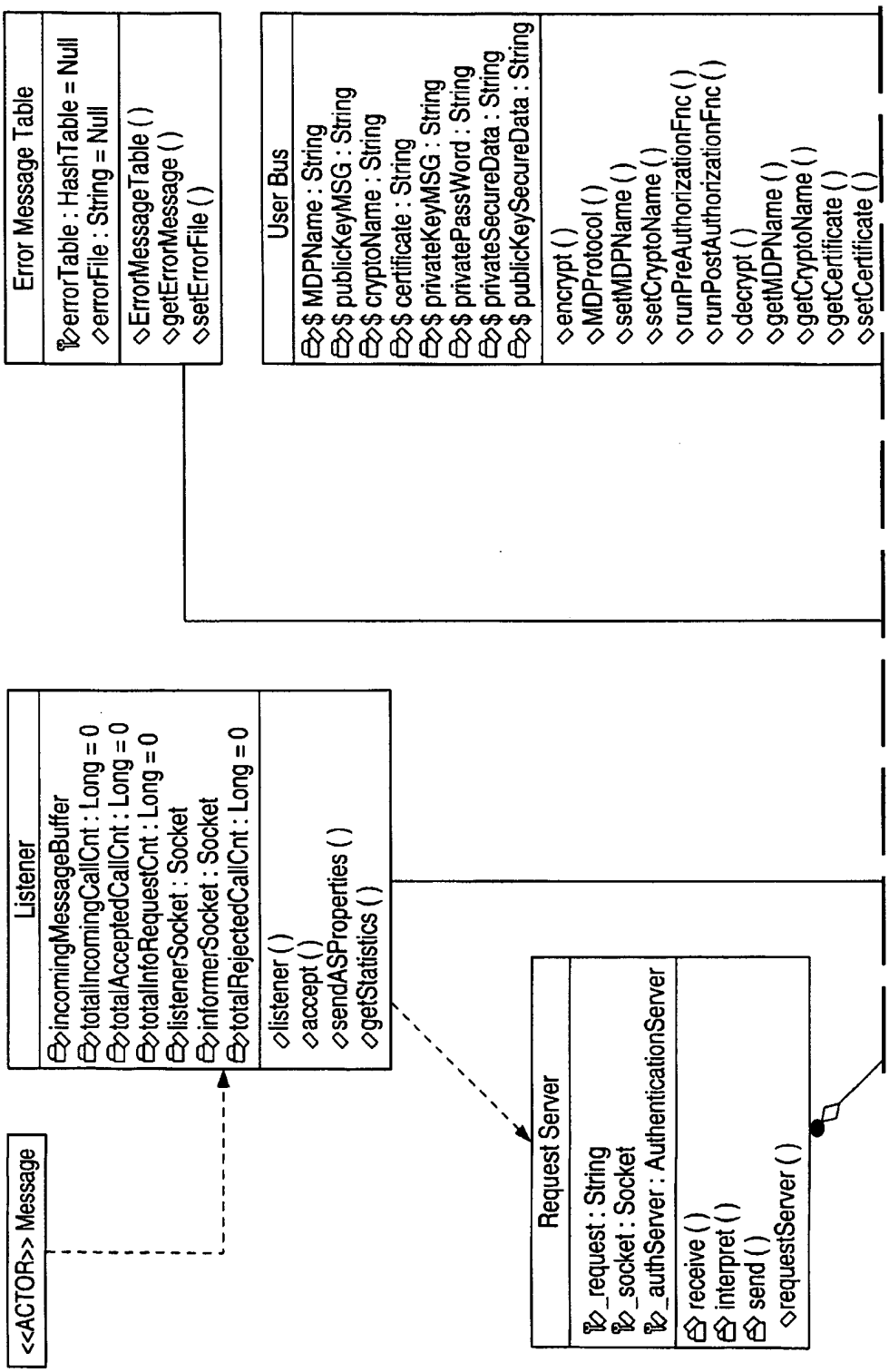

AUTHORIZATION SERVER OBJECT DIAGRAM
Error Message Table
errorTable : HashTable = Null
errorFile : String = Null
ErrorMessageTable ( )
getErrorMessage ( )
setErrorFile ( )
User Bus
$ MDPName : String
$ publicKeyMSG : String
$ cryptoName : String
$ certificate : String
$ privateKeyMSG : String
$ privatePassWord : String
$ privateSecureData : String
$ publicKeySecureData : String
encrypt ( )
MDProtocol ( )
setMDPName ( )
setCryptoName ( )
runPreAuthorizationFnc ( )
runPostAuthorizationFnc ( )
decrypt ( )
getMDPName ( )
getCryptoName ( )
getCertificate ( )
setCertificate ( )
Listener
incomingMessageBuffer
totalIncomingCallCnt : Long = 0
totalAcceptedCallCnt : Long = 0
totalInfoRequestCnt : Long = 0
listenerSocket : Socket
informerSocket : Socket
totalRejectedCallCnt : Long = 0
listener ( )
accept ( )
sendASProperties ( )
getStatistics ( )
<<ACTOR>> Message
Request Server
_request : String
_socket : Socket
_authServer : AuthenticationServer
receive ( )
interpret ( )
send ( )
requestServer ( )

FIG. 8A

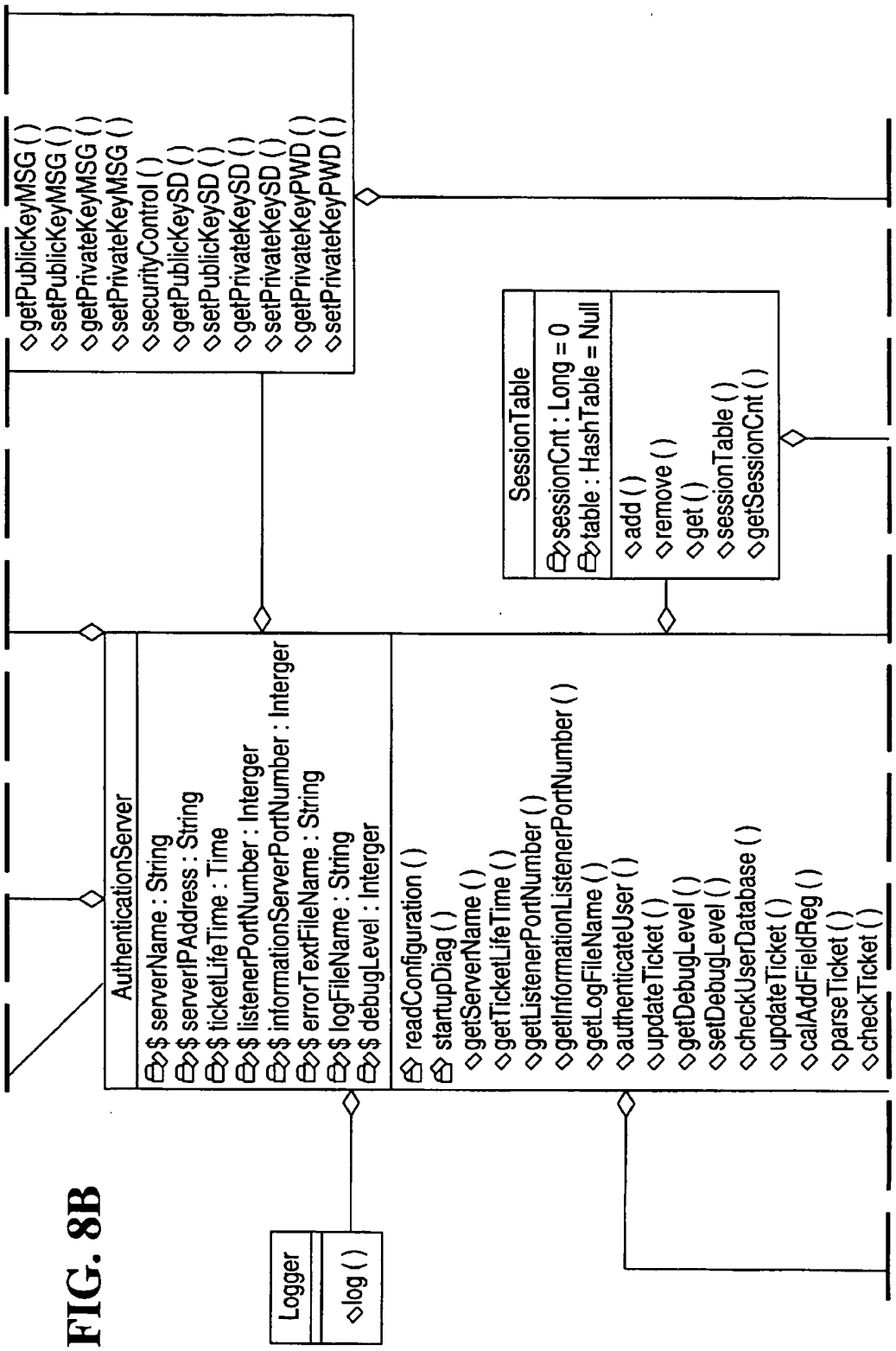

getPublicKeyMSG ( )
setPublicKeyMSG ( )
getPrivateKeyMSG ( )
setPrivateKeyMSG ( )
securityControl ( )
getPublicKeySD ( )
setPublicKeySD ( )
getPrivateKeySD ( )
setPrivateKeySD ( )
getPrivateKeyPWD ( )
setPrivateKeyPWD ( )
SessionTable
sessionCnt : Long = 0
table : HashTable = Null
add ( )
remove ( )
get ( )
sessionTable ( )
getSessionCnt ( )
AuthenticationServer
$ serverName : String
$ serverIPAddress : String
$ ticketLifeTime : Time
$ listenerPortNumber : Interger
$ informationServerPortNumber : Interger
$ errorTextFileName : String
$ logFileName : String
$ debugLevel : Interger
readConfiguration ( )
startupDiag ( )
getServerName ( )
getTicketLifeTime ( )
getListenerPortNumber ( )
getInformationListenerPortNumber ( )
getLogFileName ( )
authenticateUser ( )
updateTicket ( )
getDebugLevel ( )
setDebugLevel ( )
checkUserDatabase ( )
updateTicket ( )
calAddFieldReg ( )
parseTicket ( )
checkTicket ( )
Logger
log ( )

FIG. 8B

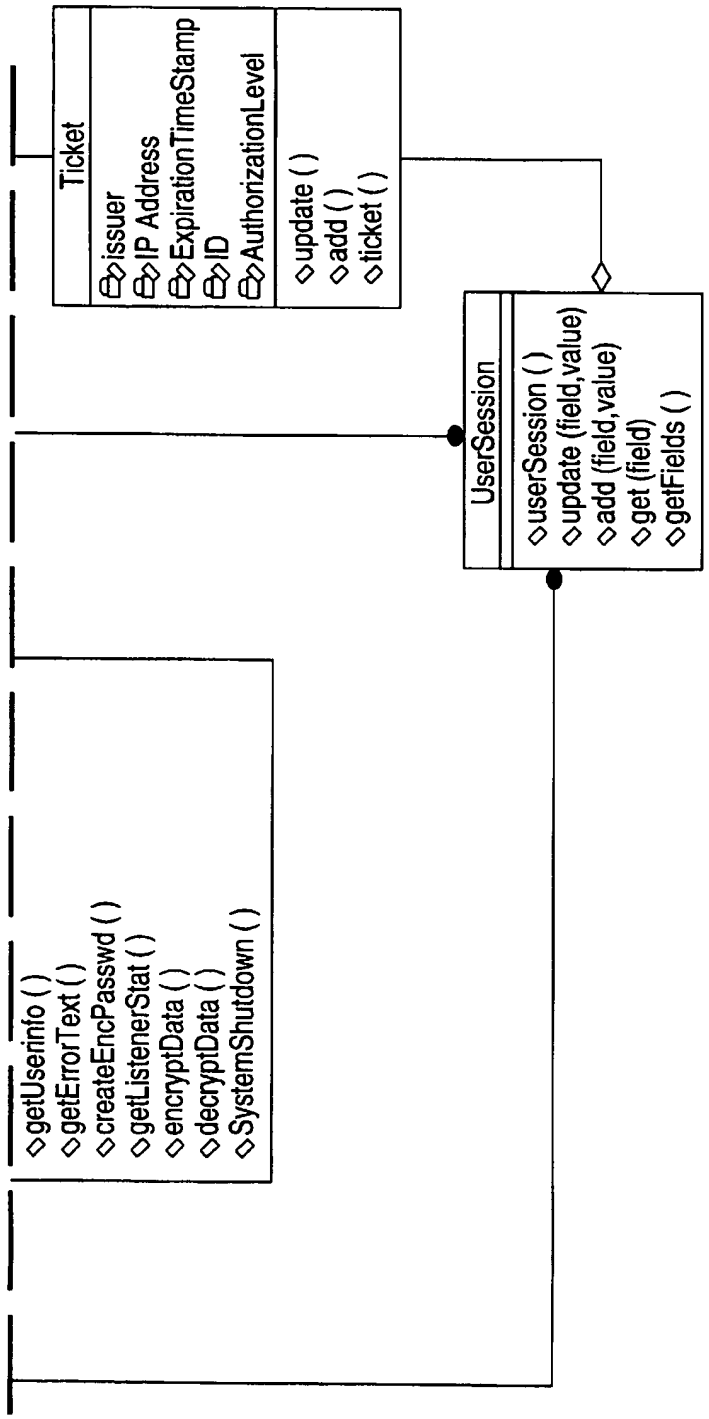

Ticket
issuer
IP Address
ExpirationTimeStamp
ID
AuthorizationLevel
update ( )
add ( )
ticket ( )
UserSession
userSession ( )
update (field,value)
add (field,value)
get (field)
getFields ( )
getUserinfo ( )
getErrorText ( )
createEncPasswd ( )
getListenerStat ( )
encryptData ( )
decryptData ( )
SystemShutdown ( )

FIG. 8C

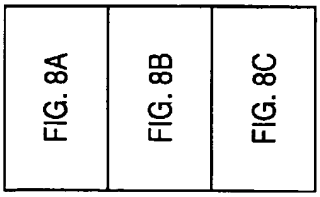


FIG. 8A
FIG. 8B
FIG. 8C

FIG. 8

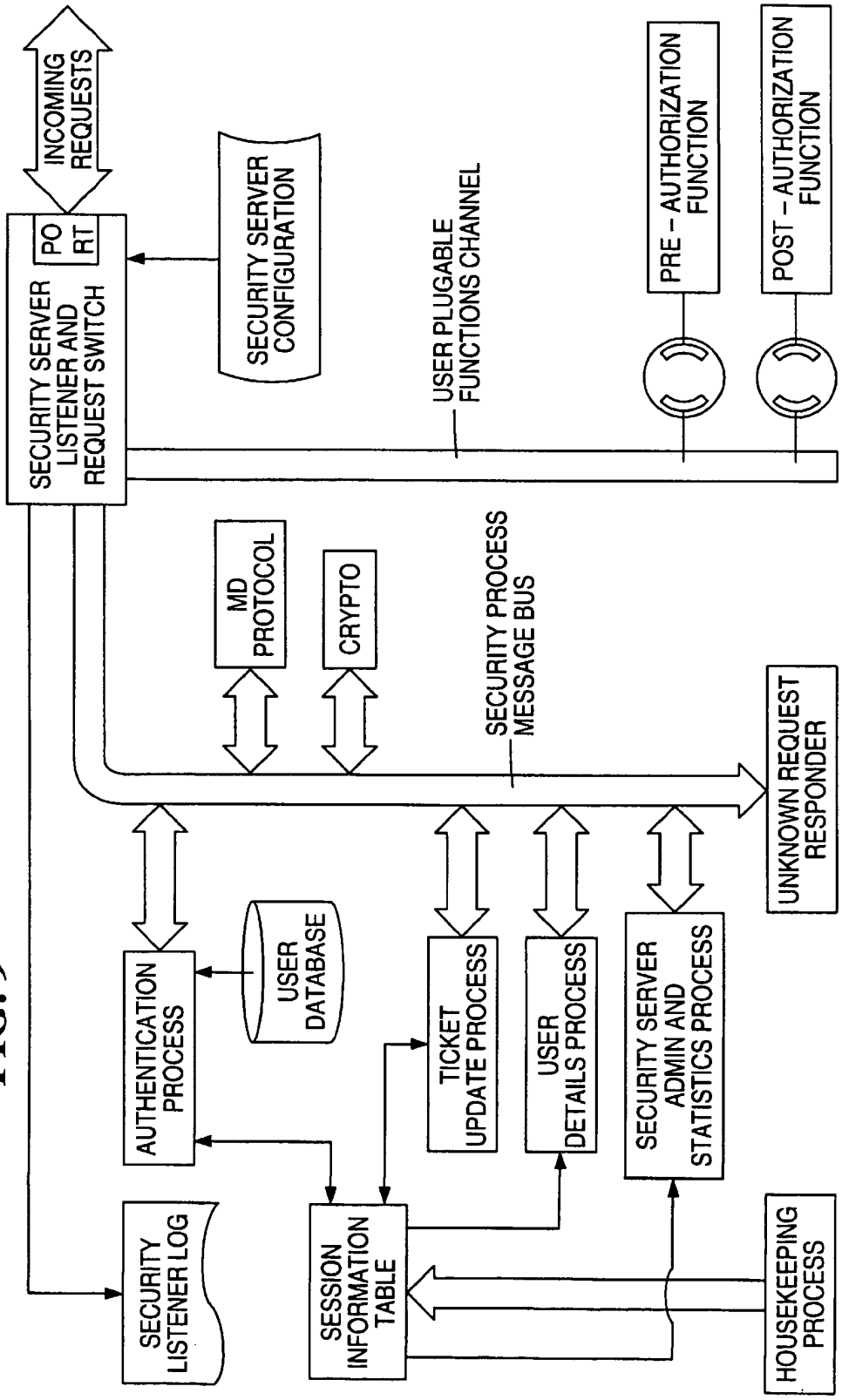

INCOMING REQUESTS
SECURITY SERVER LISTENER AND REQUEST SWITCH
PO RT
SECURITY SERVER CONFIGURATION
USER PLUGABLE FUNCTIONS CHANNEL
PRE – AUTHORIZATION FUNCTION
POST – AUTHORIZATION FUNCTION
MD PROTOCOL
CRYPTO
SECURITY PROCESS MESSAGE BUS
UNKNOWN REQUEST RESPONDER
AUTHENTICATION PROCESS
USER DATABASE
TICKET UPDATE PROCESS
USER DETAILS PROCESS
SECURITY SERVER ADMIN AND STATISTICS PROCESS
SECURITY LISTENER LOG
SESSION INFORMATION TABLE
HOUSEKEEPING PROCESS

FIG. 9

FIG. 10A


authenticationServer : AuthenticationServer
logger : Logger
sessionTable : SessionTable
userSession
ticket : Ticket
userBus : UserBus
errorMessageTable : Error Message Table
SYSTEM STARTUP INITIATION FROM SYSTEM ADMIN.
SYSTEMINITIATION PHASE STARTED
1: log (String)
2: readConfiguration ( )
3: log (String)
4: setErrorFile
5: log (String)
6: ErrorMessageTable ( )
7: log (String)
8: calAddFieldReq ( )
9: log (String)
10: SessionTable ( )
11: log (String)
12: setMDPName (String)
13: log (String)
14: setCryptoName (String)
15: log (String)

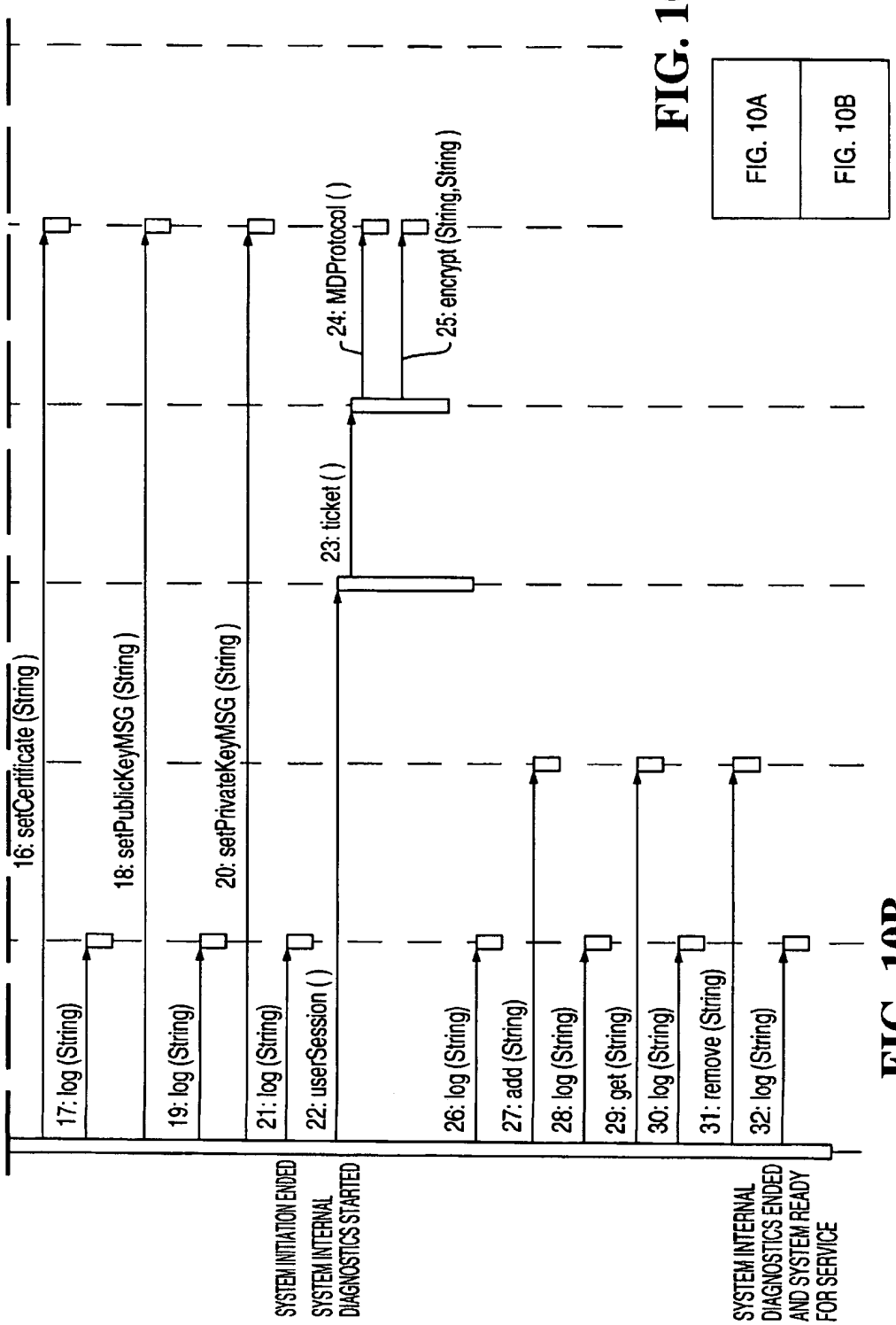


FIG. 10

FIG. 10B

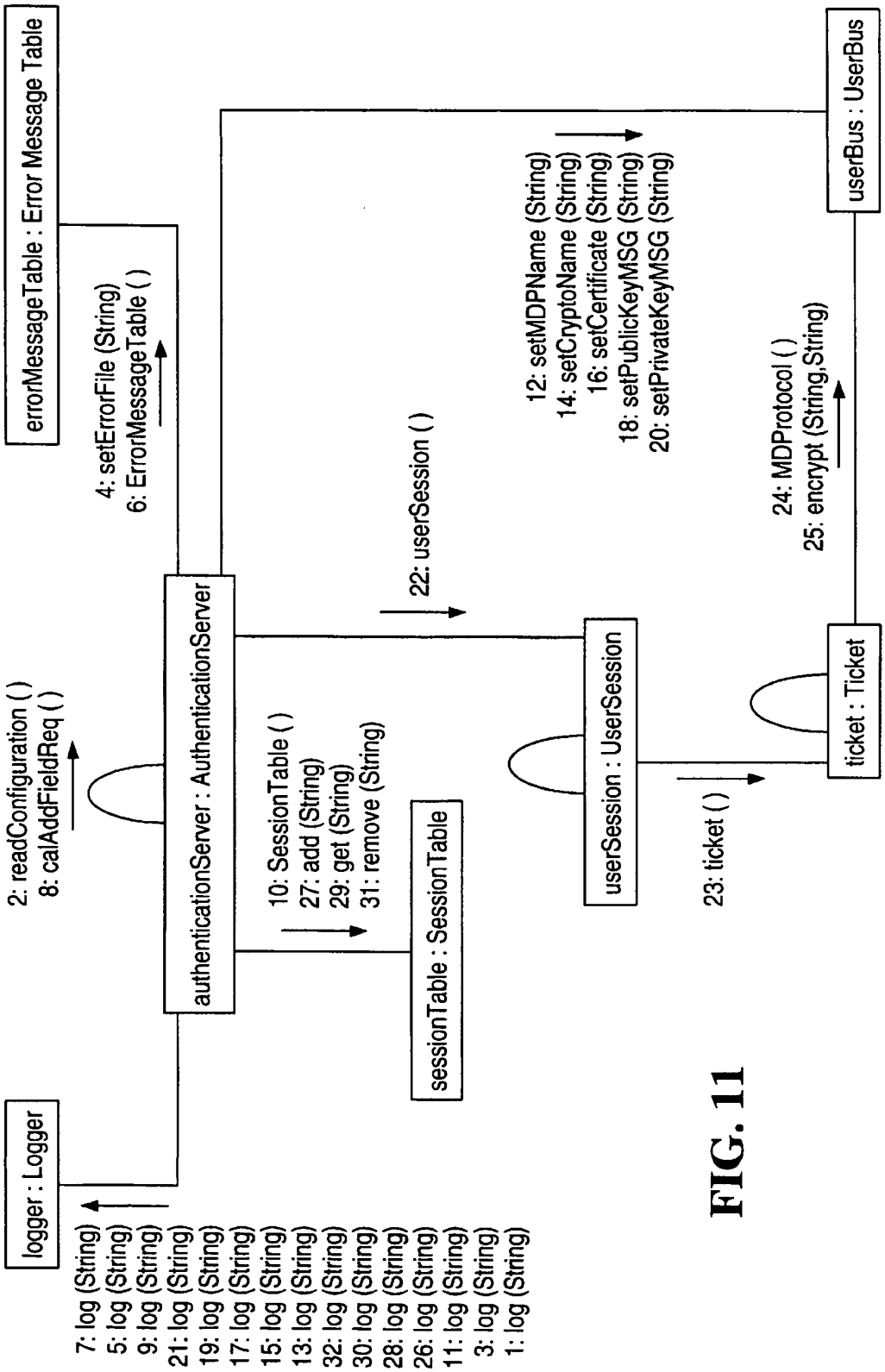

errorMessageTable : Error Message Table
userBus : UserBus
logger : Logger
authenticationServer : AuthenticationServer
sessionTable : SessionTable
userSession : UserSession
ticket : Ticket
4: setErrorFile (String)
6: ErrorMessageTable ( )
2: readConfiguration ( )
8: calAddFieldReq ( )
10: SessionTable ( )
27: add (String)
29: get (String)
31: remove (String)
22: userSession ( )
12: setMDPName (String)
14: setCryptoName (String)
16: setCertificate (String)
18: setPublicKeyMSG (String)
20: setPrivateKeyMSG (String)
24: MDProtocol ( )
25: encrypt (String,String)
23: ticket ( )
7: log (String)
5: log (String)
9: log (String)
21: log (String)
19: log (String)
17: log (String)
15: log (String)
13: log (String)
32: log (String)
30: log (String)
28: log (String)
26: log (String)
11: log (String)
3: log (String)
1: log (String)

FIG. 11

FIG. 12A

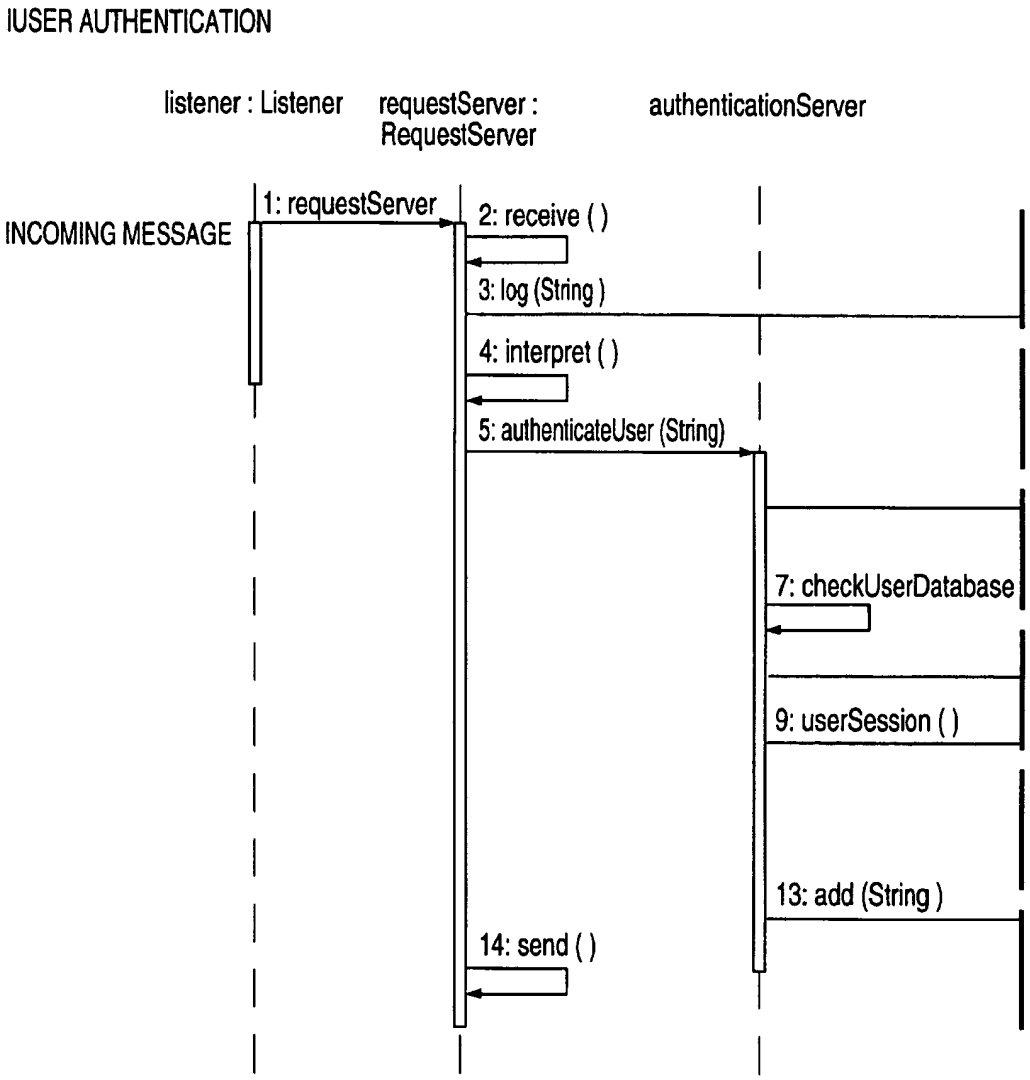


FIG. 12

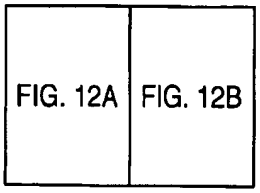


FIG. 12B

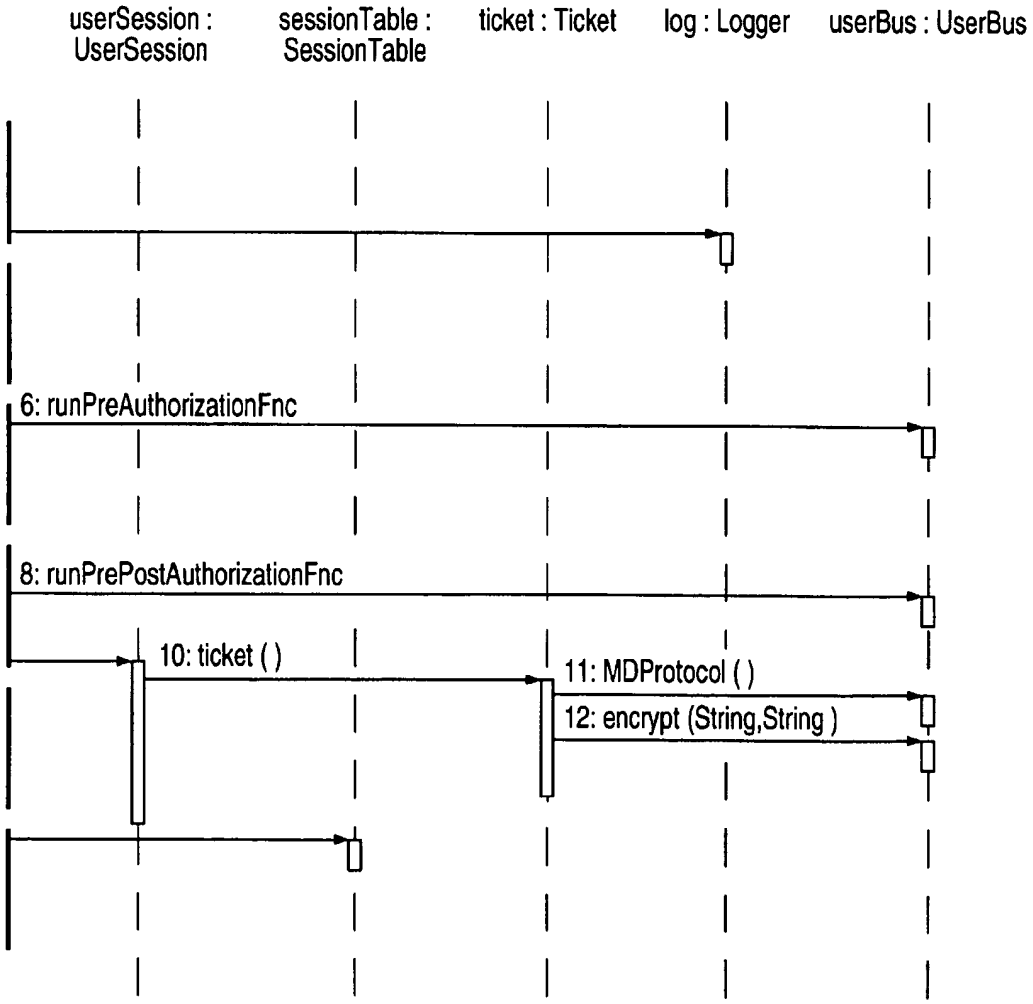
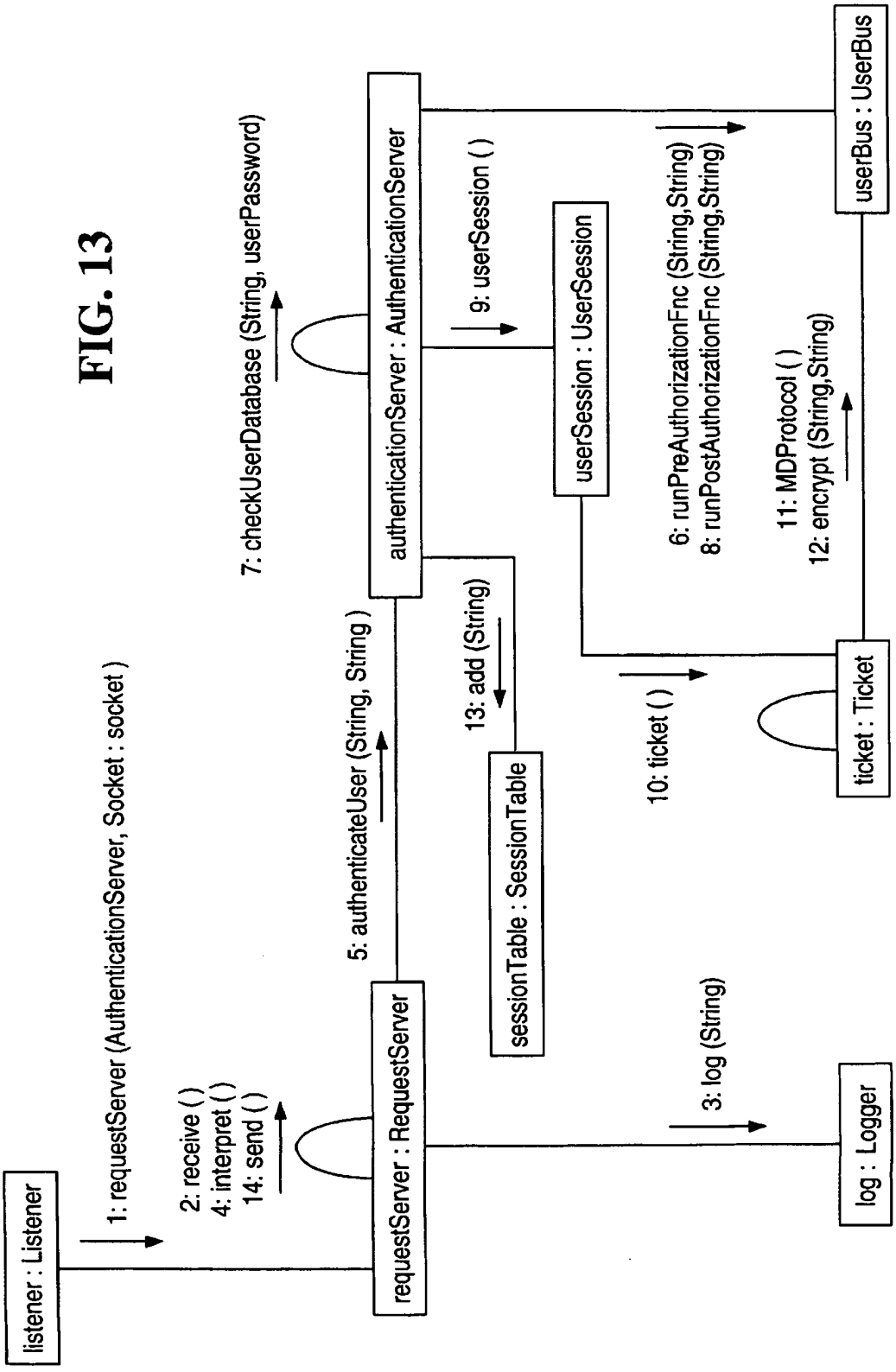

userSession : UserSession
sessionTable : SessionTable
ticket : Ticket
log : Logger
userBus : UserBus
6: runPreAuthorizationFnc
8: runPrePostAuthorizationFnc
10: ticket ( )
11: MDProtocol ( )
12: encrypt (String,String)

FIG. 13


listener : Listener
requestServer : RequestServer
authenticationServer : AuthenticationServer
sessionTable : SessionTable
userSession : UserSession
ticket : Ticket
log : Logger
userBus : UserBus
1: requestServer (AuthenticationServer, Socket : socket)
2: receive ( )
4: interpret ( )
14: send ( )
5: authenticateUser (String, String)
7: checkUserDatabase (String, userPassword)
9: userSession ( )
13: add (String)
10: ticket ( )
6: runPreAuthorizationFnc (String,String)
8: runPostAuthorizationFnc (String,String)
11: MDProtocol ( )
12: encrypt (String,String)
3: log (String)

ELECTRONIC TICKETING, AUTHENTICATION AND/OR AUTHORIZATION SECURITY SYSTEM FOR INTERNET APPLICATIONS

TECHNICAL FIELD

The present invention relates to user authentication and/or authorization of data communications and, more particularly, to data communication over a network that securely maintains user authentication and/or authorization throughout the network.

BACKGROUND ART

Many Internet protocols and applications are designed to serve large public user groups. Because of this, Internet Servers were designed to serve their community in a stateless manner. One request to the server has no relationship to the previous or next request. All requests are independent, rather than considered as part of a user "session" to that server. This approach simplified server activity to service many requests from many users, without having to establish and track sessions for each user. However, the approach introduces a new problem to solve; user privacy and security.

In a network environment, security issues such as communication channel integrity and privacy, user authentication, and user authorization exist. Communication between two end points in a network has to be guarded against outside intervention (i.e., High Voltage noise, Lightning or Human). Security affording protection against this kind of intervention is commonly referred to as communication channel integrity and privacy.

Channel integrity and privacy precautions against "natural" events and are typically handled by communication protocols. Algorithms have been developed over the years to perfect and solve these "natural" events and have been proven effective through many years of usage. However, when introducing a channel integrity and privacy problem, such as Human intervention, the reliability of these algorithms deteriorates. Protocol level controls typically do not encrypt data, enabling human intervenors to change Cyclic Residency Control (CRC) information and any information on an open transmission channel. Hence, any user sensitive data (for example, credit card numbers or other private user information) traveling on the Internet can be obtained by any human intervenor.

In an effort to resolve this problem, Web Technology providers architected Secure Socket Layer (SSL). SSL is the product residing between Web applications and the Communication Protocol Layer. SSL provides data encryption, server authentication and message integrity for TCP/IP connections. This effectively handles protecting the privacy and integrity of data traveling over the Internet.

User authentication is defined as "determining the true identity of a user or an object attempting to access a system." Any non-public system has to have an authentication system in order to filter and identify users from one another. However, Web servers do not typically keep track of the user identity throughout the duration of that users visit to the site. For complete security, the user identity must be provided with each request made of the Web server. This may be accomplished by having the user "log on" for each new request, or by conducting a behind the scenes "re-authentication" of the user for each request. These techniques are, however, inconvenient for the user and/or time consuming for the application.

User authorization involves determining what types of activities are permitted for an authenticated user or object. Authorization is generally grouped into two categories: (1) Data Set Authorization (typically controlled by the application), and (2) Function Set Authorization (typically controlled by the operating system).

Based on the foregoing, we have determined that web user "authentication" must first be accomplished before optionally following with user "authorization". Hence, efficiency may be increased if "authentication" for each "authorization" request is eliminated.

SUMMARY OF THE INVENTION

To overcome the above-identified disadvantages and shortcomings of the prior art, it is a feature and advantage of the present invention to transmit data over a system, such as the world wide web, in a more secure and efficient manner.

It is another feature and advantage of the present invention to provide user authentication information which is maintained throughout transmission over a system, such as the world wide web.

It is another feature and advantage of the present invention to provide user authorization information in addition to the authentication information, enabling the user to gain access to system resources provided, for example, over the world wide web.

According to one aspect of the invention, a computer program memory stores computer instructions, generating an electronic ticket used for verifying user authorization to provide secure data communication over a system. The computer instructions generate a data packet based on authorization information, hash the information in the data packet to produce a hash number, encrypt the hash number to prevent unauthorized alteration of the information in the data packet and concatenate the data packet and encrypted hash number to produce a ticket. The ticket may then be transmitted in a non-secure environment and a user may be authorized based on the validation of the integrity of the information in the ticket.

In one embodiment of the invention, MD5 protocol is used to hash the information in the data packet.

In another embodiment of the invention, a private key is used to encrypt the hash number.

In another embodiment of the invention, the is identification information includes issue host name, client IP address, expiration date and time and authorization level. User extension information may also be provided.

Another aspect of the invention provides a method for securing data communication over a system, including generating an electronic ticket used for verifying user authorization to provide secure data communication over a system, producing a signature by hashing at least the authentication information, encrypting the signature, concatenating the information in the data packet with the encrypted signature, and transmitting the ticket over the system in a non-secured environment. A user is authorized to access system resources upon validating the integrity of the information in the ticket having been transmitted in the non-secured environment.

In one embodiment of the invention, MD5 protocol is used to hash the information in the data packet.

In another embodiment of the invention, a private key is used to encrypt the signature.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and the advantages thereof, reference is now made to

the following descriptions taken in conjunction with the accompanying drawings in which like numbers designate like parts, and in which:

FIG. **1** is a block diagram which illustrates a computer system upon which an embodiment of the invention may be implemented.

FIG. **2** is a diagram of a network within which the present invention may be implemented.

FIG. **3** is an exemplary diagram of a ticket in an embodiment of the present invention.

FIG. **4** is an exemplary diagram of an "eticket" in an embodiment of the present invention.

FIG. **5** is a flow diagram of a method of generating an "eticket" in an embodiment of the present invention.

FIG. **6** is a flow diagram of a high level flow authentication and authorization process of the present invention.

FIG. **7** is a ticket flow diagram of a high level flow authentication and authorization process of the present invention.

FIG. **8** is a class object diagram of the authorization server.

FIG. **9** is a diagram of a high level architecture of the authentication server.

FIGS. **10***a* and **10***b* are interaction diagrams notation of the start-up process of the authentication server.

FIG. **11** is an object message diagram representing the start-up process of the authentication server.

FIG. **12** is an interaction diagram notation of a user authentication attempt.

FIG. **13** is an object message diagram representing a user authentication attempt.

NOTATIONS AND NOMENCLATURE

The detailed descriptions which follow may be presented in terms of program procedures executed on a computer or network of computers. These procedural descriptions and representations are the means used by those skilled in the art to most effectively convey the substance of their work to others skilled in the art.

A procedure is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. These steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared and otherwise manipulated. It proves convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like. It should be noted, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities.

Further, the manipulations performed are often referred to in terms, such as adding or comparing, which are commonly associated with mental operations performed by a human operator. No such capability of a human operator is necessary, or desirable in most cases, in any of the operations described herein which form part of the present invention; the operations are machine operations. Useful machines for performing the operation of the present invention include general purpose digital computers or similar devices.

The present invention also relates to apparatus for performing these operations. This apparatus may be specially constructed for the required purpose or it may comprise a general purpose computer as selectively activated or reconfigured by a computer program stored in the computer. The procedures presented herein are not inherently related to a particular computer or other apparatus. Various general purpose machines may be used with programs written in accordance with the teachings herein, or it may prove more convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these machines will appear from the description given.

BEST MODE FOR CARRYING OUT THE INVENTION

A method and apparatus for information discovery and visualization are described. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

Hardware Overview

FIG. **1** is a block diagram which illustrates a computer system **100** upon which an embodiment of the invention may be implemented. Computer system **100** includes a bus **102** or other communication mechanism for communicating information, and a processor **104** coupled with bus **102** for processing information. Computer system **100** also includes a main memory **106**, such as a random access memory (RAM) or other dynamic storage device, coupled to bus **102** for storing information and instructions to be executed by processor **104**. Main memory **106** also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor **104**. Computer system **100** further includes a read only memory (ROM) **108** or other static storage device coupled to bus **102** for storing static information and instructions for processor **104**. A storage device **110**, such as a magnetic disk or optical disk, is provided and coupled to bus **102** for storing information and instructions.

Computer system **100** may be coupled via bus **102** to a display **112**, such as a cathode ray tube (CRT), for displaying information to a computer user. An input device **114**, including alphanumeric and other keys, is coupled to bus **102** for communicating information and command selections to processor **104**. Another type of user input device is cursor control **116**, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor **104** and for controlling cursor movement on display **112**. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), which allows the device to specify positions in a plane.

The invention is related to the use of computer system **100** to discover and visualize information according to a configurable information model. According to one embodiment of the invention, information discovery and visualization is provided by computer system **100** in response to processor **104** executing sequences of instructions contained in main memory **106**. Such instructions may be read into main memory **106** from another computer-readable medium, such as storage device **110**. However, the computer-readable medium is not limited to devices such as storage device **110**.

For example, the computer-readable medium may include a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punch cards, paper tape, any other physical medium with patterns of holes, a RAM, a PROM, an EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave embodied in an electrical, electromagnetic, infrared, or optical signal, or any other medium from which a computer can read. Execution of the sequences of instructions contained in main memory **106** causes processor **104** to perform the process steps previously described. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

Computer system **100** also includes a communication interface **118** coupled to bus **102**. Communication interface **108** provides a two-way data communication coupling to a network link **120** that is connected to a local network **122**. For example, communication interface **118** may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface **118** may be a local area network (LAN) card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface **118** sends and receives electrical, electromagnetic or optical signals which carry digital data streams representing various types of information.

Network link **120** typically provides data communication through one or more networks to other data devices. For example, network link **120** may provide a connection through local network **122** to a host computer **124** or to data equipment operated by an Internet Service Provider (ISP) **126**. ISP **126** in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" **128**. Local network **122** and Internet **128** both use electrical, electromagnetic or optical signals which carry digital data streams. The signals through the various networks and the signals on network link **120** and through communication interface **118**, which carry the digital data to and from computer system **100**, are exemplary forms of carrier waves transporting the information.

Computer system **100** can send messages and receive data, including program code, through the network(s), network link **120** and communication interface **118**. In the Internet example, a server **130** might transmit a requested code for an application program through Internet **128**, ISP **126**, local network **122** and communication interface **118**. In accordance with the invention, one such downloaded application provides for information discovery and visualization as described herein.

The received code may be executed by processor **104** as it is received, and/or stored in storage device **110**, or other non-volatile storage for later execution. In this manner, computer system **100** may obtain application code in the form of a carrier wave.

Network Overview

Referring to FIG. **2**, depicted is a network **200** within which the present invention may be implemented. A web server **220** according to one embodiment of the present invention gathers information dynamically from one or more data sources, which may be located at different servers and have incompatible formats, structures the information into an object-oriented, information model, and outputs the information for the user according to an associated visual representation. The information model and the visual representation are defined by human operators according to their own needs, purposes, and preferences as part of the configuration of the server. Multiple information models and visual representations may be defined for any server.

A user may access the web server **220** by executing a web browser at client **210**. Web browsers are well-known in the art, and are readily available from such corporations as Netscape Communications Corp. and Microsoft Corp. In order to access the web server **220**, the user at client browser **210** activates a hyperlink having a URL (Uniform Resource Locator) of the following form:

```
http://www.server.com/query.pl?Class=Seed&View=
  Paradigm                                    [TABLE 1]
```

In the exemplary URL, the network address of the web server **220** is specified as "www.server.com" and the portion of the URL after the question mark (?) hold user specified parameters. The Class and Seed parameters, as explained in more detail hereinafter, indicate an object about which a user intends to discover information. The object is visualized according a paradigm specified by the Paradigm parameter, also explained in more detail hereinafter.

When the hyperlink is activated, the web server **220** receives a request to initiate an information discovery session, specified by parameters embedded in the URL. In response, the web server **220** gathers information from one or more data sources. The data sources can have incompatible formats, e.g. web page, relational database, spreadsheet, text file, etc. The data sources can be stored at a plurality of sites, for example, locally with respect to the web server **220**, such as a hard disk at local storage **222**, or externally at another site in the network, e.g. at mainframe **230**. In fact, the data source can even be another, remote information discovery web server **240**.

Electronic Ticketing Architecture

Two server models are used to authenticate a user and maintain that authentication and authorization throughout, for example, a web site visit. As shown in FIG. **6**, the first is the Authentication Server **350** (or Intelligent Ticketing Server), and the second is the function based Authorization Server **360**. The Authentication Server **350** receives authentication information from a user and generates an eticket **310** (discussed in more detail below), and the Authorization Server **360** uses the information in the eticket **310** to check authorization functionality. In fact, authorization or access rights of the user are carried as a part of the eticket **310**. The two servers need not exist in the same hardware platform, and they are not tightly coupled. Hence, the eticket **310** provides trusted authentication information about the user to the authorization server **360**, and the authorization server **360** can perform its authority check functions.

Generally, the eticket **310** is signed with an industry standard public Message Digest signature (MD Protocol), and is protected by a public key encryption system. Hence, anyone who knows the public key may validate the eticket **310** without having to communicate with the Authentication Server **350**. This reduces communication between distributed server applications.

FIG. **3** is an exemplary diagram of a ticket in an embodiment of the invention. Extendible ticketing architecture "eticket" is based on public encryption algorithms. The

eticket **310** has two functional parts. The first part contains the ticket framework **302** and the second part is a "seal" (signature) **306** which protects the eticket **310** content from alterations. Information contained in the eticket **310** are divided by a field separator, for example, a pipe "|" symbol, into separate fields of data.

For example, ticket framework **302** may contain the following information:

(1) Ticket Issuer: Ticket issuer host name (i.e. BillBlasterSrv),

(2) Client IP Address: XXX.XXX.XXX.XXX format (i.e. 65.192.217.6),

(3) Expiration Date and Time: YY-MM-DD HH:MM:SS (97-10-01 10:30:15), and

(4) User Authorization Level: Single String Text (i.e. AccMngO1).

FIG. **4** is an exemplary diagram of an eticket **310** in an embodiment of the present invention. Ticket extension (i.e., optional user extension **304**) is accomplished by adding any additional information to the eticket **310**. For example, the eticket **310** may contain additional "custom" information, such as user age, credit card number and zip code. When user extension **304** information is inserted into the eticket **310** framework, the data will reside between ticket framework **302** and the Public Key Signature "PKS" **306** fields. Hence, all ticket extension information will automatically be included in the Message Digest Protocol "MDP" and will affect the signature of the eticket **310**.

With the eticket **310** architecture, the following security issues in, for example, the world wide WEB environment are addressed:

(1) IP Spoofing,

(2) Time controlled session,

(3) Ticket alterations,

(4) Authorization Levels, and

(5) User dependent persistent data storing.

FIG. **5** is a flow diagram of a method of generating an eticket in an embodiment of the present invention. In step S1, the authentication server **350** receives authentication information from a user and generates an eticket **310**. At this stage of the process, the eticket **310** (excluding the public key signature field) contains two "fields" of information—(1) the ticket framework **302** and (2) the user extension **304**. The ticket framework **302** contains information such as ticket issuer server name, client IP address, expiration date and time, and user authorization, as previously described. The user extension **304** contains information such as user age, credit card number and zip code, also previously described. The information in the eticket **310** (excluding the public key signature field) is then, for example, optionally hashed using, for example, MD5 protocol, and encrypted with a public key encryption system in step S2, generating a hash number in step S3. Of course, other hashing or encryption techniques may also be used. The hash number is representative of the specific information contained in the current eticket **310**. The hash number generated in step S3 is then encrypted using a standard private key encryption system in step S4. Encrypting the hash number with a private key encryption system prevents anyone without knowledge of the private key from viewing the hash number. In step S5, the eticket **310** and the encrypted hash number are concatenated to generate the completed "eticket" framework **310**. Hence, the completed "eticket" framework **310** includes three "fields" of information—(1) the ticket framework **302**, (2) the user extension **304**, and (3) the public key signature (encrypted hash number **306**).

FIG. **6** is a flow diagram of a high level authentication process and flow authorization of the present invention. In step T1, a user provides authentication information (such as user ID and password) to, for example, a station terminal, and requests an operation. This "input" step inherently provides a secure operation, as the user is the only one who has access to the authentication information at this stage of the process. The user authentication information is then passed from the station terminal to the authentication server **350** for user validation. The authentication server **350** is the core of the system, and serves to coordinate processes of the system. Hence, the authentication server **350** properties are stored in a server property file, and a ReadConfiguration method will access the property file when the system initiates (see, for example, FIGS. **10***a* and **10***b*). The file includes, for example, the following properties:

ServerName:	Authentication Server Name
IP_Address:	Authentication Server IP Address
TicketLifeTime:	Maximum life cycle of one single operation
ErrorTextFileName:	Full path of Error Message file
LogFileName:	Full path of Authentication Server log file
Debug:	System Debug Level 0- None
ListenerPortNumber:	Listener Port Number
InfoPortNumber:	Information Server Port Number
MessageDigestName:	Message Digest Protocol Name (MD-2, ND-5 etc.)
CryptoName:	Algorithm used for Message Protection
publicKeyMSG:	Public Key for ticket signature
privateKeyMSG:	Private Key for ticket signature
privateKeyPassWord:	Private Key for password encryption
privateKeySecureData:	Private Key for data security requests (Encryption)
publicKeySecureData:	Public Key for data security requests (Decryption)
certificate:	Server Certificate

In step T2, the authentication server **350** validates the user authentication information and generates (or issues) an eticket **310** (including ticket framework **302**, user extension **304**, and public key signature **306**), as described above, in step T3. The eticket **310** may then be securely passed from server to server, without the user having to "reauthenticate" each time a new server accesses the information. That is, a user does not have to "re-authenticate" all of the authentication information each time a new server is accessed. Rather, the eticket **310** provides the information (ticket framework **302** and user extension **304**) to the server. This information is passed in a secure manner as a result of the public and private key encryption systems (see example below).

The eticket **310** is returned to the client browser **210** when the user requests a service. The request (including the eticket **310**) is sent to the web server **220** or **240**. In step T4, a web script checks the eticket **310** to determine whether the eticket **310** is authentic or invalid. The information in the eticket **310** must be rehashed. If the authorization server **360** has access to the hashing technique and public key used to encrypt the ticket hash information **304**, the authorization server **360** can rehash and decrypt the eticket signature **306**. If the hashing technique and public key operate to properly decrypt and rehash the eticket **310**, then the information stored in the eticket **310** is determined to be valid. However,

it should be noted that the authorization server **360** only has the ability to access the information in the eticket **310**, it does not generally have the ability to alter the contact of the eticket **310**. Hence, the information in the eticket **310** remains secure. In order for the information in the eticket **310** to be altered, the server must have access to the private key used to encrypt the public key signature **306** (see example below).

Once the eticket **310** has been rehashed and decrypted, the operation request will be returned to the web script in step **T8**. If the eticket **310** is invalid, an error message is generated, then the user operation request is rejected by the web server **220** or **240** in step **T5**. If, on the other hand, the eticket **310** is valid, then the user request may continue to be processed by the authorization server **360**.

The authorization control process begins in step **T6**, and the eticket **310** is passed to the authorization server **360** with the requested operation, in step **T7**. As previously stated, there is no need for the user to "re-authenticate" herself at the authorization server **360** because the eticket **310** includes all of the necessary authentication information. In order for the authorization server **360** to determine whether or not the user is authorized for the specific user authorization level, the information in the eticket **310** is used to determine if the function is authorized for this user (step **T9**). If the requester is authorized and the request is granted, it is executed in step **T11** and the results are returned to the requester. If, on the other hand, the requester is not authorized and the request is denied, the user request is rejected in step **T10**.

An example of processing the secured eticket **310** follows. Assume that the eticket comprises: (1) a four field ticket framework (fields **1–4**), (2) a four field user extension (fields **5–8**), and (3) a public key signature (field **9**).

	eticket								
	1	1	1	1	1	1	1	1	8
Field	1	2	3	4	5	6	7	8	9

Referring to FIG. **5**, we can step through the "eticket" process. In this example, each of the four fields in the "eticket" framework **302** and user extension **304** include data represented by the number "1" in step **S1**. The Message Digest/Hash is represented by a summation (Σ) algorithm (equated to, or exemplary of, the MD5 protocol or other hashing algorithm). Hence, to calculate the Message Digest/Hash, a summation algorithm is implemented using all eight fields of data in step **2**. Thus, the summation of all eight fields of data results in the numerical value "8". This "8" represents the Message Digest/Hash. Once the Message Digest/Hash has been generated, it may be encrypted in step **S3** using a private key. The encrypted signature is then generated in step **S4**. In this example, the encrypted public key signature of "8" is represented by "X". In step **S5**, the encrypted public key signature is concatenated with the ticket framework and user extension information to generate the secure "eticket", as follows.

	secure ticket								
	1	1	1	1	1	1	1	1	X
Field	1	2	3	4	5	6	7	8	9

FIG. **7** is a ticket flow diagram of a high level flow authorization and authentication process of the present invention. This figure is a simplified version of the diagram illustrated in FIG. **6**. A simplified "flow" of the eticket **310** on the system is as follows. A user submits a sign-on request in step **V1**. The sign-on request is submitted to the intelligent ticketing server (authentication server) **350**, and the user is authenticated in step **V2**. An eticket **310** is generated by the intelligent ticketing server **350** using authentication information supplied by the user upon sign-on. The eticket **310** is passed to a web server, for example, **220** or **240**, in step **V3** the eticket **310** is then returned to the user. When the user requests a function, the request and eticket **310** are sent to the web server **220** or **240**, and the server sends the requested eticket **310** in step **V4** to function based authorization server **360**. An authentication and authorization check is performed in step **V5**, and a determination is made as to whether or not the user is authorized to perform the requested function. If the user is not authorized, then the request made by the user is denied. If, on the other hand, it is determined that the user is authorized, then the request made by the user, for example a request for protected documents, will be validated and returned in step **V6** to the web server **220** or **240**.

FIG. **8** is a class object diagram of the authorization server. FIG. **9** is a diagram of a high level architecture of the authentication server. FIGS. **10A** and **10B** are interaction diagrams notation of the start-up process of the authentication server. FIG. **11** is an object message diagram representing the start-up process of the authentication server. FIG. **12** is an interaction diagram notation of a user authentication attempt. FIG. **13** is an object message diagram representing a user authentication attempt.

This invention will provide a much more confined user and a world wide WEB based application system by providing a user session concept with the "eticket" architecture. This ticketing architecture will tie the user browser **210** to the Internet Server **240** (or application). It uses all industry standard security components like public key encryption or message/text unique signature system (also known as message digest protocols). Hence, it becomes nearly impossible to alter any information in the "eticket". At the same time it is very easy to validate and honor the ticket for authorization purposes even in a distributed server application.

The many features and advantages of the invention are apparent from the detailed specification, and thus, it is intended by the appended claims to cover all such features and advantages of the invention which fall within the true spirit and scope of the invention. Further, since numerous modifications and variations will readily occur to those skilled in the art, it is not desired to limit the invention to the exact construction and operation illustrated and described, and accordingly, all suitable modifications and equivalents may be resorted to, falling within the scope of the invention.

What is claimed:

1. A computer program memory, comprising said memory storing computer instructions to use an electronic ticket, said electronic ticket being adapted for verifying user authorization and providing secure data communication over a system, said electronic ticket including a ticket framework and a signature field, said computer instructions including:

generating said electronic ticket using a data packet having information including user authorization information;

receiving a request at a plurality of web servers, said request including a service request and said electronic ticket;

determining authorization of said request received at each of said web servers by determining authorization of said electronic ticket; and

fulfilling said service request if said electronic ticket is authorized and not fulfilling said service request if said electronic ticket is not authorized.

2. The computer program memory of claim **1**, wherein the electronic ticket further comprises extension information.

3. The computer program memory of claim **2**, wherein the extension information comprises:

an age of the user, a credit card number of the user, and a zip code of the user.

4. The memory of claim **1**, wherein said data packet used in generating said electronic ticket further includes user authentication information and wherein the memory further includes the computer instructions:

determining authentication of said request received at said web server by determining authentication of said electronic ticket; said determining authentication instruction is performed prior to said determining authorization instruction.

5. The memory of claim **4**, wherein said generation instruction uses private key encryption and said determining authentication instruction uses public key decryption.

6. The memory of claim **1**, wherein said generation instruction uses private key encryption and said authorization instruction uses public key decryption.

7. The memory of claim **1**, wherein one web server is a web-based application server.

8. A method for using an electronic ticket for verifying user authorization and providing secure data communication over a system, comprising the steps of;

a first step of generating said electronic ticket using a data packet having information including user authorization information;

a second step of receiving a request at a plurality of web servers, said request including a service request and said electronic ticket;

a third step of determining authorization of said electronic ticket received at each of said web servers; and

a fourth step of fulfilling said service request if said electronic ticket is authorized and not fulfilling said service request if said electronic ticket is not authorized.

9. The method of claim **8**, wherein the electronic ticket further comprises extension information.

10. The method of claim **9**, wherein the extension information comprises:

an age of the user, a credit card number of the user, and a zip code of the user.

11. The method of claim **8**, wherein said data packet used in said first step of generating said electronic ticket further includes user authentication information and wherein the method further comprises:

a step of determining authentication of said request received at said web server by determining authentication of said electronic ticket; said determining authentication step is performed prior to said third step of determining authorization of said electronic ticket.

12. The method of claim **8**, wherein said first step of generating said electronic ticket uses private key encryption and said determining authentication step uses public key decryption.

13. The method of claim **8**, wherein said first step of generating said electronic ticket uses private key encryption and said third step of determining authorization uses public key decryption.

14. The method of claim **8**, wherein said web server is a web-based application server.

15. A computer program memory, comprising said memory storing computer instructions to generate an electronic ticket, said electronic ticket being adapted for verifying user authorization and providing secure data communication over a system, said electronic ticket including a ticket framework and a signature field, said computer instructions including:

providing a data packet having unencrypted information based on at least authorization information, said authorization information includes extension information;

producing a signature by hashing at least the authorization information;

encrypting only the signature to prevent unauthorized alteration of the signature;

concatenating the unencrypted information in the data packet with the encrypted signature to produce the electronic ticket.

16. The computer program memory of claim **15**, wherein the extension information comprises:

an age of the user, a credit card number of the user, and a zip code of the user.

17. The memory of claim **15**, wherein said data packet further includes user authentication information.

18. The memory of claim **15**, wherein said encrypting instruction uses private key encryption.

19. A method for generating an electronic ticket used for verifying user authorization to provide secure data communication over a system, comprising the steps of:

a first step of providing a data packet having unencrypted information based on at least authorization information, said authorization information includes extension information;

a second step of producing a signature by hashing at least the authorization information;

a third step of encrypting only the signature to prevent unauthorized alteration of the signature;

a fourth step of concatenating the unencrypted information in the data packet with the encrypted signature to produce the electronic ticket.

20. The method of claim **19**, wherein the extension information comprises:

an age of the user, a credit card number of the user, and a zip code of the user.

21. The method of claim **19**, wherein said data packet further includes user authentication information.

22. The method of claim **19**, wherein said third step of encrypting uses private key encryption.

* * * * *